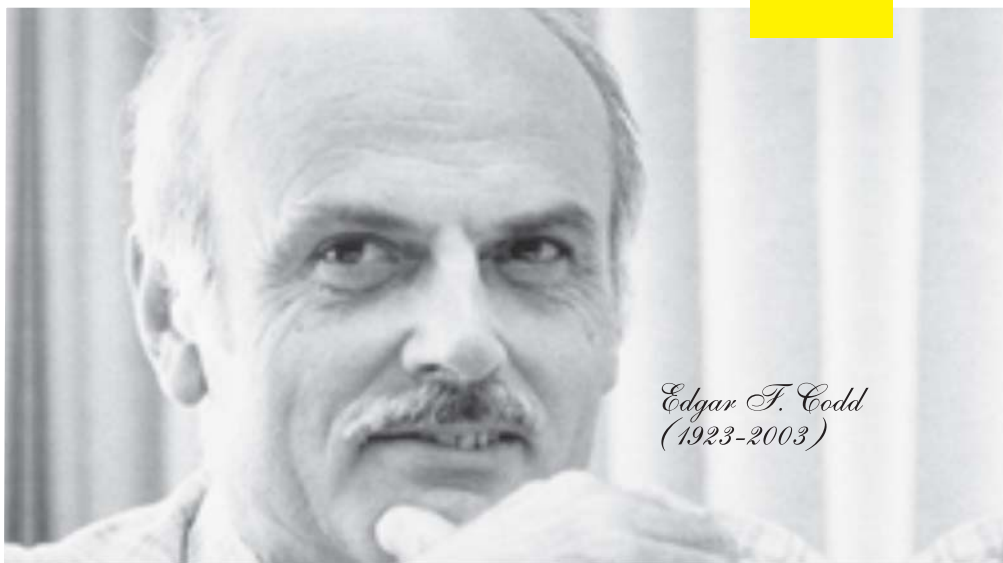


Ministerul Educației, Cercetării și Tineretului

MARIANA PANȚIRU

Manual pentru clasa a 12-a



*Edgar F. Codd
(1923-2003)*

INFORMATICĂ

Editura
ALL



MARIANA PANȚIRU

INFORMATICĂ

Manual pentru clasa a 12-a

Editura
ALL



Această carte în format digital (e-book) intră sub incidența drepturilor de autor și a fost creată exclusiv pentru a fi citită utilizând dispozitivul personal pe care a fost descărcată. Oricare alte metode de utilizare, dintre care fac parte împrumutul sau schimbul, reproducerea integrală sau parțială a textului, punerea acestuia la dispoziția publicului, inclusiv prin intermediul Internetului sau a rețelelor de calculatoare, stocarea permanentă sau temporară pe dispozitive sau sisteme – altele decât cele pe care a fost descărcată – care permit recuperarea informațiilor, revânzarea sau comercializarea sub orice formă a acestui text, precum și alte fapte similare, săvârșite fără acordul scris al persoanei care deține drepturile de autor, sunt o încălcare a legislației referitoare la proprietatea intelectuală și vor fi pedepsite penal și/sau civil în conformitate cu legile în vigoare.

Informatică – Manual pentru clasa a XII-a

Mariana PANȚIRU

Copyright © 2007, 2012 ALL EDUCATIONAL

ISBN 978-973-684-808-7

Manualul a fost aprobat prin Ordinul ministrului Educației, Cercetării și Tineretului nr. 1262/58 din 6.06.2007 în urma evaluării calitative și este realizat în conformitate cu programa analitică aprobată prin Ordin al ministrului Educației și Cercetării nr. 5959 din 22.12.2006.

Referenți: **prof. gr. I Cornelia Ivașc**
prof. gr. I Carmen Losonczy

Coperta colecției: **Alexandru Novac**

Redactor: **Daniela Slavu**

Tehnoredactare: **Gabriel Iancu**

Editura ALL

Bd. Constructorilor nr. 20A, et. 3,
sector 6, cod 060512, București
Tel.: 021 402 26 00
Fax: 021 402 26 10

Distribuție:

021 402 26 30; 021 402 26 33

Comenzi:

comenzi@all.ro

www.all.ro

CUPRINS

Capitolul 1 Organizarea datelor în baze de date	5
Informații și date. Organizarea datelor	5
Operații generale asupra unei structuri de date	7
Evoluția metodelor de organizare a datelor	9
Baze de date și sisteme de gestiune a datelor	13
Capitolul 2 Baze de date relaționale	22
Concepte specifice bazelor de date relaționale	22
Proiectarea bazelor de date relaționale: forme normale	27
Capitolul 3 Introducere în mediul FoxPro	33
FoxPro: prezentare generală	33
Interfața produsului Visual FoxPro	35
Configurarea mediului FoxPro	36
Moduri de lucru	36
Gestiunea fișierelor	40
Capitolul 4 Tipuri de date în Visual FoxPro și operații specifice	43
Tipul numeric	43
Tipul șir de caractere	44
Tipul dată calendaristică	45
Tipul logic	46
Gestiunea variabilelor	46
Comenzi de lucru cu variabilele	48
Macrosubstituție și expresii nume	49
Capitolul 5 Crearea tabelelor	52
Definirea structurii unui tabel	52
Modificarea structurii conceptuale a unui tabel	54
Editarea câmpurilor unui tabel	54
Popularea (încărcarea) tabelului cu date	55
Deschiderea și închiderea unui tabel	55
Filtrarea structurii sau selectarea câmpurilor	56
Filtrarea articolelor	56
Capitolul 6 Vizualizarea, căutarea și sortarea datelor	59
Vizualizarea conținutului unei tabele	59
Vizualizarea structurii unei tabele	60
Căutare secvențială și poziționare în baza de date	60
Sortarea și duplicarea unei tabele	61
Capitolul 7 Actualizarea datelor	65
Adăugarea articolelor	65
Ștergerea articolelor	66
Modificarea sau corectarea datelor	67
Actualizarea interactivă a tabelelor	67
Lucrul cu câmpurile de tip Memo	73
Lucrul cu câmpurile de tip General	75
Capitolul 8 Indexare și căutare rapidă	79
Indexarea tabelelor	79
Căutarea rapidă și poziționarea în tabela indexată	82
Capitolul 9 Relaționarea tabelelor	85
Tipuri de relații	85
Crearea și ștergerea unei relații	85
Capitolul 10 Transferul de date între tabele Visual FoxPro și alte structuri	90
Importarea și exportarea datelor din și către tablouri de memorie	90
Importarea și exportarea datelor din și către alte tipuri de fișiere	92
Capitolul 11 Prelucrări statistice și financiare	94
Numărarea articolelor: COUNT	94
Însumarea valorilor unor câmpuri: SUM	94
Calculul mediei aritmetice: AVERAGE	95
Diverse calcule statistice și financiare: CALCULATE	95
Totalizarea valorilor: TOTAL	96
Funcții financiare	96
Recapitulare	99

Capitolul 12 Programarea clasică în FoxPro	106
Structura alternativă și comanda IF...ENDIF	106
Structura selectivă și comanda DO CASE...ENDCASE	106
Structura repetitivă și comanda DO WHILE...ENDDO	106
Structura repetitivă și comanda SCAN...ENDSCAN	107
Structura repetitivă și comanda FOR...ENDFOR	107
Leșiri forțate: LOOP și EXIT	107
Proceduri utilizator	111
Funcții utilizator	111
Activitatea de depanare a programelor	116
Sfaturi pentru depistarea și eliminarea erorilor	117
Strategii de depanare	119
Utilizarea instrumentului Debugger	119
Capitolul 13 Operații cu baze de date în Visual FoxPro	124
Descrierea operațiilor cu baze de date prin comenzi	124
Gestiunea interactivă a bazei de date. Database Designer	125
Crearea rapidă a unei baze de date cu Database Wizard	126
Deschiderea și activarea unei baze de date	127
Operații asupra unei tabele incluse în baza de date	128
Proiectarea interactivă a tabelei prin Table Designer	131
Fixarea relațiilor persistente între tabelele unei baze de date	135
Proceduri stocate	138
Declanșatoare	139
Proiectarea regulilor de integritate a bazei de date	140
Capitolul 14 Interogarea bazelor de date	145
Definirea interogărilor prin comanda SELECT	145
Proiectarea vizuală a interogărilor	149
Proiectarea fișierelor View	158
Capitolul 15 Comunicarea aplicației Visual FoxPro cu alte aplicații	166
Proiectarea fișierelor vedere cu date la distanță	166
Proiectarea rapidă a vederilor cu Remote View Wizard	167
Proiectarea paginilor web pentru vizualizarea datelor pe internet	168
Proiectarea paginilor de căutare pe Internet	169
Proiectarea și transmiterea documentelor prin e-mail. Utilitarul Mail Merge Wizard	170
Recapitulare	172
Capitolul 16 Elemente de programare orientată spre obiecte	175
Proprietățile definatorii ale claselor	177
Comenzile FoxPro necesare programării obiectelor	177
Capitolul 17 Proiectarea formularelor	181
Generatorul de formulare (Form Designer)	182
Proiectarea unui formular prin Quick Form	187
Proiectarea vizuală a obiectelor de interfață	190
Proiectarea interfeței folosind Form Wizard	212
Capitolul 18 Afișarea datelor sub formă de rapoarte	217
Rapoarte	217
Etichete	229
Grafice	231
Capitolul 19 Proiectarea meniurilor	234
Generatorul de meniuri Menu Builder	234
Capitolul 20 Proiecte și aplicații	238
Conducerea aplicației printr-un program monitor	238
Organizarea aplicației sub formă de proiecte. Project Builder	239
Generarea aplicațiilor executabile cu Project Manager	241
Realizarea dischetelor de distribuție	243
Capitolul 21 Dezvoltarea profesională în domeniul IT	246
Capitolul 22 Teme opționale	248
Protecția bazelor de date	248
Tehnici avansate de gestiune a structurii conceptuale a unei tabele	251
Capitolul 23 Dezvoltarea unei aplicații informatice	255
Aspecte teoretice	255
Proiectul final	261
Studii de caz	266
Răspunsuri, comentarii, indicații	279

Organizarea datelor în baze de date

- *Informații și date, organizarea datelor*
- *Operații generale asupra unei structuri de date*
- *Evoluția metodelor și tehnicilor de organizare a datelor*
- *Baze de date și SGBD*

INFORMAȚIA, alături de energie și materii prime, joacă un rol esențial în producția fizică de bunuri materiale și constituie esența tuturor activităților intelectuale, de la conducere, educație, artă, viață colectivă și individuală, la întreținerea patrimoniului de cunoștințe al umanității.

Informații și date. Organizarea datelor

Activitatea umană, în cele mai diverse forme ale sale, a fost întotdeauna caracterizată prin entități factice, exprimate fie sub formă de valori numerice, fie ca percepții sau observații nenumerice făcute de ființele umane sau de mașini. Aceste entități factice independente și neevaluate sunt numite DATE. Datele obținute în cadrul activităților productive, de conducere, de cercetare, educaționale, artistice constituie un material informațional brut care poate fi *evaluat, ordonat și prelucrat*, având în vedere diferite obiective. În urma acestui proces de transformare a datelor se obțin INFORMAȚII, care reprezintă o interpretare a datelor în raport cu anumite situații particulare sau cu înțelegerea de către mintea umană în general. Informațiile constituie baza raționamentelor, experimentărilor imaginate de mintea umană în scopul obținerii de noi CUNOȘTINȚE.



Figura 1-1: Transformarea datelor în informații, apoi în cunoștințe

Producerea informațiilor susține anumite acțiuni umane cu finalitate practică, creând totodată un fond informațional utilizabil pentru generarea de noi informații și cunoștințe.

INFORMAȚIA este definită ca „o comunicare susceptibilă de a produce cunoștințe“, are un caracter semantic și de noutate, de aport la cantitatea de cunoștințe a celui care o primește. Informația se referă la obiecte, persoane, procese, fenomene, locuri, situații, condiții etc. O informație trebuie să fie utilă, precisă, completă, fără prea multe amănunte care să facă mesajul greu de interpretat, să sosească la timp pentru luarea unor decizii. Eschil zicea că înțelept nu este cel ce știe lucruri *multe* ci lucruri *utile*. O altă trăsătură fundamentală a informației este subiectivitatea sa. Ceea ce este o informație pentru o persoană poate să nu însemne nimic pentru alta.

DATELE constituie materializarea, reprezentarea simbolică a informațiilor (prin semne, litere, cifre) într-o formă convențională (scrisă, vorbită, luminoasă, semne grafice, desene) convenabilă unei comunicări. Datele descriu ce s-a întâmplat prin număr sau cuvânt, nu furnizează vreo judecată sau interpretare și nici baza pentru acțiune. Datele reprezintă „materia primă“ din care, după o serie de operațiuni efectuate de către oameni sau echipamente, se obțin informații.

Informația reprezintă semnificația ce poate fi desprinsă dintr-un ansamblu de date, pe baza asociațiilor dintre acestea, este deci partea utilizabilă dintr-un ansamblu de date. Informațiile furnizează răspunsuri la întrebări de genul „cine“, „unde“, „când“. Datele sunt exprimate prin atribut și valoare. Exemple: produs=„imprimantă“, cantitate=20, prețul=150Lei. Informația se exprimă prin propoziții. Exemplu: „S-au vândut 20 imprimante la prețul de 150 Lei“.

Informația presupune înțelegerea relațiilor dintre date

Pentru obținerea informațiilor se desfășoară mai multe activități, grupate sub numele de PROCES INFORMAȚIONAL, dintre care amintim:

- *identificarea, culegerea sau înregistrarea datelor;*
- *pregătirea datelor* – cuprinde ansamblul activităților de filtrare, codificare, clasificare, conversie, selectare;
- *prelucrarea datelor;*
- *memorarea datelor* pe diverse medii în vederea refolosirii lor;
- *comunicarea sau raportarea informațiilor* către utilizatorii lor.

SISTEMUL INFORMATIC este definit ca ansamblul mijloacelor (tehnice, umane, financiare) și al metodelor specifice prin care se asigură desfășurarea procesului informațional, având ca obiectiv obținerea informațiilor necesare conducerii unei activități.

COLECȚIA DE DATE reprezintă un ansamblu de date care se referă la același fenomen, obiect sau situație.

Între componentele unei colecții de date, ca și între colecții, pot fi identificate sau, eventual, pot fi introduse **relații** care să determine pe mulțimea respectivă o anumită **structură**, adică un anumit mod de ordonare care să faciliteze prelucrarea. Relațiile pot fi unidirecționale sau bidirecționale. O relație unidirecțională poate determina obținerea unei valori sau a mai multora din componenta legată plecând numai de la o componentă (părinte); o relație bidirecțională determină obținerea aceluiași date sau valori plecând de la ambele colecții. Componentele structurii pot fi individualizate și selectate prin **poziția** pe care o ocupă în structură, în raport cu ordinea specificată sau, mai comod, prin **numele componentei**.

O colecție de date pe care s-au definit anumite relații și căreia îi este specific un anumit mecanism de selecție și identificare a componentelor constituie o **STRUCTURĂ DE DATE**. Mecanismul de selecție al unei structuri de date este implementat de obicei în programele de prelucrare a datelor respective, la nivelul sistemului de operare, al sistemului de gestiune sau al programelor aplicative.

Există două mari **tipuri de acces**:

- *Accesul secvențial* presupune parcurgerea tuturor datelor situate înaintea datei care trebuie prelucrată;
- *Accesul direct* presupune un mecanism prin care se poate determina direct poziția datei necesare prelucrării.

Operații generale asupra unei structuri de date

- creare (memorarea pe suport a datelor);
- consultare (acces la elementele structurii pentru prelucrarea valorilor);
- actualizare (inserare, ștergere sau corecție a valorilor);
- copiere (duplicarea structurii, în general pe un alt suport);
- ventilare (desfacerea structurii în două sau mai multe structuri);
- fuzionare (combinarea a două sau mai multe structuri);
- sortarea (aranjarea elementelor structurii după anumite criterii).

Componentele unei structuri de date pot să fie de același tip (structura este **omogenă**) sau de tipuri diferite (structură **neomogenă**).

Operațiile (operatorii) aplicate asupra unei structuri de date pot să-i afecteze valorile și/sau structura. Dacă o structură de date își modifică structura în timp, este numită **dinamică**. Spre deosebire de acestea, structurile **stative** au același număr de componente și în aceeași ordine pe tot parcursul existenței lor.

Definirea structurilor de date necesare într-o aplicație este o activitate complexă, care condiționează în mare măsură necesarul de memorie, viteza de prelucrare și uneori chiar efortul de proiectare și implementare. Pentru realizarea ei trebuie să se țină seama de o serie de factori, dintre care:

- volumul datelor;
- operațiile de prelucrare și frecvența lor (o atenție deosebită trebuie acordată actualizării: dacă este frecventă, trebuie aleasă o structură la care performanțele accesului nu scad esențial datorită modificărilor structurii);
- indicele de activitate pe operații – se definește ca raport între numărul de componente ale structurii utilizate într-o operație și numărul de componente explorate pentru aceasta. El determină timpul de acces. Dacă indicele este mare (peste 0,8), atunci accesul poate fi secvențial. Valorile mici ale indicelui implică un acces direct.
- durata de viață a structurii;
- utilizarea rațională a spațiului de memorie (comprimare, blocare, segmentare, combinarea diferitelor forme de reprezentare);
- complexitatea programării;
- asigurarea integrității datelor (alegerea structurii care permite protecția împotriva distrugerilor accidentale și posibilitatea refacerii datelor).



sarcini de laborator

1. Catalogul clasei conține mai multe date.

- Identificați aceste date și faceți o listă a atributelor lor. Separați datele care sunt elementare și datele care sunt calculate. Pentru fiecare scrieți natura valorilor posibile (numere sau șiruri de caractere).
- Ce informații pot fi cerute de conducerea școlii, de părinți, de elevi?
- Identificați diferite structuri care să faciliteze obținerea rapidă a informațiilor. Care sunt prelucrările?

De exemplu, o colecție ar putea să rețină numele elevului, numele părinților și adresa de acasă, o altă colecție ar putea să rețină numărul matricol al elevului, totalul absențelor motivate și nemotivate, o colecție ar putea să conțină notele (data când a fost luată, elevul, obiectul) etc.

2. Accesați site-ul www.olimpiade.ro și faceți o listă a datelor memorate acolo. Ce informații s-ar putea obține?

3. Căutați pe Internet și accesați un site pentru o companie de transporturi aeriene. Faceți o listă cu activitățile desfășurate, datele memorate, informațiile posibile obținute din prelucrarea acestor date

4. Căutați și vizitați site-ul unei librării on-line. Cum sunt prezentate cărțile? Este dată și o imagine a copertii sau a autorilor? Care sunt informațiile pe care le puteți obține din pagina web?

5. Faceți un mic proiect, cercetând activitatea bibliotecarului școlii. Luați interviuri, cercetați documentele cu care lucrează bibliotecarul, observați direct ce se întâmplă la bibliotecă.

- Primul raport va detalia activitățile, cine le desfășoară, când, cu ce periodicitate, ce decizii se iau.
- Al doilea raport conține un inventar al documentelor, fiecare cu numele său, dacă este standard sau nu, dacă este primit din exterior sau staționar sau trimis mai departe către alte organizații, câte exemplare conține, cine verifică și semnează, cât timp necesită redactarea documentului.
- Realizați machetele tuturor documentelor.
- Identificați datele de pe documente și tipurile lor. Faceți o grilă informațională, plasând pe linii atributul și pe coloane numele documentului. La intersecție marcați cu X prezența datei în documentul respectiv. Separați datele de informații!
- Analizați rapoartele, răspunzând la întrebări de felul: toate activitățile se desfășoară automatizat/manual? Există date redundante? Care sunt prelucrările datelor (sortări, calcule, comparații etc.)?

Evoluția metodelor de organizare a datelor

Organizarea datelor a parcurs mai multe etape, pornind de la fișiere, apoi sisteme de fișiere, până la baze de date.

Fișiere pe aplicații

Primele forme de organizare a datelor au fost fișierele secvențiale pe bandă magnetică. Odată cu folosirea suportului adresabil – discul magnetic – au apărut fișierele cu acces direct (indexate, selective). Gestiunea fișierelor era asigurată printr-un sistem specializat, numit SGF (sistem de gestiune a fișierelor) existent în orice sistem de operare. A fost o soluție specifică anilor 1960-1970, dar mai este întâlnită și astăzi în cazul aplicațiilor economice care folosesc limbaje clasice precum Cobol.

Fișierul este o structură omogenă din punctul de vedere al conținutului și al prelucrării. Organizarea în fișiere are un nivel fizic și un nivel logic. Fișierul este format din înregistrări fizice – care formează unitatea de transfer între memoria externă și cea internă. O înregistrare fizică poate conține mai multe înregistrări logice (formate din câmpuri), care reflectă punctul de vedere al utilizatorului.

Programele utilizatorilor trebuie să conțină în mod explicit referiri la suportul, forma de organizare și modul de acces al fișierului de lucru.

Altfel spus, *datele aveau sens și puteau fi accesate numai din aplicațiile-program.*

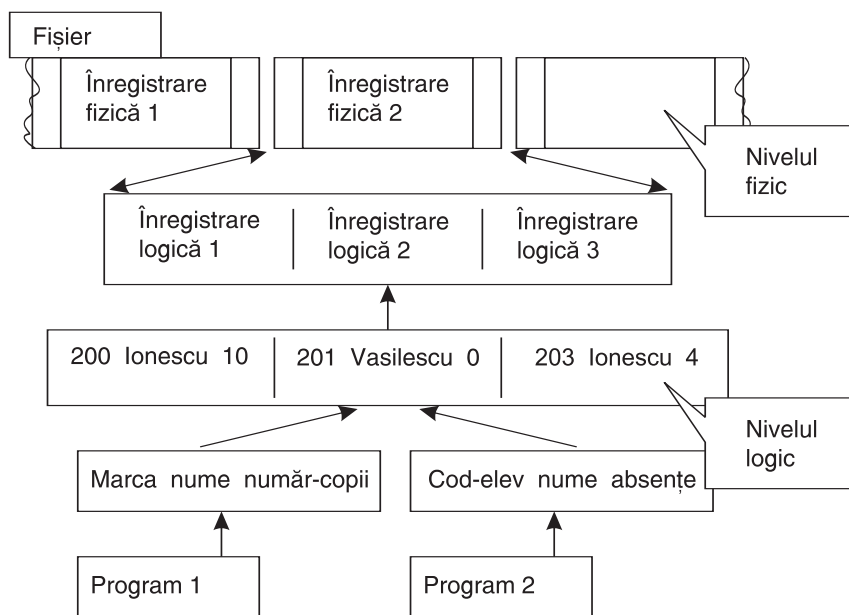


Figura 1-2: Exemplet de acces la date

Prezența elementelor de descriere a datelor din fișiere în programele de aplicații este un mare dezavantaj, pentru că orice modificare a structurii și organizării datelor pe suportul fizic conduce la modificarea tuturor aplicațiilor care le folosesc.

Pe de altă parte, în anii 1960-1970 fiecare aplicație informatică dintr-o instituție era proiectată într-o manieră individualistă și viza automatizarea unei anumite activități. În vederea creșterii vitezei de răspuns, aplicația își aducea în fișierele proprii toate datele de care avea nevoie, ignorând faptul că, poate, aceleași date erau culese, selectate și prelucrate și de alte aplicații. Apare *fenomenul de redundanță a datelor* și efectele sale nedorite: creșterea nejustificată a costurilor de întreținere a fișierelor de date și, mai ales, pericolul inconsistenței datelor (datorită omiterii actualizării datelor în toate locurile unde apăreau).

Sisteme de fișiere

Pasul următor în istoricul organizărilor datelor a fost gestiunea comună a fișierelor de către toate aplicațiile informatice care aveau legături funcționale bine stabilite între ele. Apare structurarea datelor în **fișiere integrate** sau **sisteme de fișiere**.

Trecerea la sistemele de fișiere nu a rezolvat, însă, o problemă foarte importantă: *problema independenței programelor de aplicații de modul de organizare a datelor*. În organizările tradiționale pe fișiere, cunoștințele despre structura datelor și tehnicile de accesare a acestora erau incluse în programe. Odată creat fișierul, toate procedurile de prelucrare conțineau aceste caracteristici. Modificarea structurii fișierului (adăugarea unor câmpuri sau eliminarea altora) conducea la modificarea porțiunilor descriptive ale programelor, iar modificarea modului de acces (secvențial, direct) impunea revizuirea conținutului procedural al programelor. O cerință nouă se rezolva doar printr-un program nou sau prin modificarea unuia existent. Dacă luăm în considerație faptul că aproape zilnic apăreau cerințe noi, efortul de menținere a sistemului de fișiere operațional era imens. Soluția a fost *detașarea din programul de aplicații a descrierii fișierelor* și a legăturilor dintre ele. Astfel au apărut primele baze de date.

Primele baze de date; apariția SGBD-urilor

O bază de date a fost creată ca o *colecție unică*, un ansamblu de date unitar organizat și structurat, la care să aibă acces diverși utilizatori, nu neapărat programatori, pentru operații curente de adăugare de valori, corecții, extrageri de date, asemeni operațiilor dintr-o bancă obișnuită. *Declararea structurii logice a datelor se face în exteriorul aplicației, asigurând independența programelor față de structura datelor*. Apare acum și oportunitatea unui sistem standard de proceduri de gestionare a bazei, numit **Sistem de gestiune a bazei de date** (prescurtat, SGBD). Sarcinile acestui SGBD erau legate de definirea structurii datelor și a criteriilor de acces, oferirea unor măsuri de confidențialitate a datelor, oferirea posibilității accesului la date atât pentru programatori cât și pentru utilizatorii obișnuiți, neinformaticieni etc. Rămâne un punct slab însă faptul că *metoda de acces la date trebuia inclusă în programele aplicației*.

Folosirea dicționarului bazei de date; apariția funcției de administrator

La primele SGBD-uri utilizatorul trebuia să cunoască, pe lângă datele necesare aplicației proprii, și descrierea tuturor datelor din baza de date, lucru care, pe lângă greutatea în utilizare, însemna și o lipsă de confidențialitate a bazei de date.

În etapa ulterioară, preocuparea principală a fost de a degaja utilizatorul de sarcina de a cunoaște întreaga structură a bazei de date, mai ales că bazele de date au devenit foarte complexe. S-a ajuns astfel la o *gestiune independentă a structurii generale* a bazei de date, denumită *nivel virtual de organizare* a datelor. Acest lucru este realizat prin intermediul unui fișier de descriere globală a bazei de date, denumit, în general, **dicționar de date** (sau repertoriu de date sau catalog al sistemului). Lucrul cu baza de

date se derulează exclusiv prin intermediul acestui dicționar care conține informații privitoare la structurarea datelor și restricțiile îndeplinite de acestea. Astfel apare noțiunea de model conceptual al bazei de date.

Specifică acestei etape este și apariția **funcției de administrator** al bazei de date și conceperea unor proceduri speciale de securitate.

Apariția modelului relațional de baze de date

O adevărată revoluție în gestionarea bazelor de date a provocat-o modelarea structurii conceptuale a datelor sub forma tabelelor (numite *relații*), la sfârșitul anilor 1970 de către Codd, un matematician de la Centrul de cercetări din San Jose (California) al firmei IBM. Structura propusă este independentă de tipul echipamentului și software-ului de sistem pe care este implementată baza de date. Modelul relațional marchează începutul unei noi generații de SGBD-uri, la care utilizatorul nu se mai preocupă de *descrierea căilor de acces*, ca la generația precedentă, această sarcină fiind pusă *în seama sistemului*.

Folosirea bazelor de date în rețelele de calculatoare

Odată cu răspândirea rețelilor de calculatoare au fost dezvoltate diverse tehnici de comunicație, de partajare a datelor, de utilizare în comun a diferitelor componente software și hardware. Dar, pentru rețelele mari, cu sute sau mii de tranzacții, există inconvenientul duratei mari de răspuns din cauza, pe de o parte, a vitezelor relativ mici ale liniilor de comunicație între calculatoare, iar pe de altă parte a volumului mare de date de pe aceste linii. **Modelul client-server** este o tehnologie care rezolvă o serie de probleme apărute în utilizarea rețelilor de calculatoare. Presupune folosirea sistemelor informatice distribuite și reprezintă un model în care mai multe programe autonome comunică între ele.

Apărute la granița dintre două domenii aparent opuse din punctul de vedere al procesării datelor (baze de date și rețele de calculatoare), **bazele de date distribuite** se definesc ca o colecție de baze de date interconectate logic, distribuite într-o rețea de calculatoare.

Transformarea unui sistem centralizat de baze de date într-un sistem distribuit constă în scindarea bazei de date în secțiuni corespunzătoare și repartizarea acestora în nodurile rețelei. Alocarea partițiilor în nodurile unei rețele ține cont de prima regulă a distribuirii datelor, aceea că *datele vor fi distribuite numai din motive de viteză*. Plasarea datelor se face pe stația în care producerea și utilizarea este mai frecventă, astfel încât să se minimizeze transportul de date prin rețea.

Utilizatorul nu trebuie să cunoască modul de partiționare și distribuire a datelor, deoarece lucrează în același mod ca și cu o bază centralizată. Singurul lucru pe care îl cunoaște este schema conceptuală a bazei de date.

Sistemele moderne de gestiune a bazelor de date conțin elemente de **prelucrare paralelă**. Acest concept servește la reunirea în cadrul aceleiași exploatare a prelucrărilor de rutină (interogări mai simple, dar numeroase) cu prelucrările decizionale (interogări complexe ale unui volum important de date). Un SGBD trebuie să realizeze împărțirea unei interogări complexe în module executate pe mai multe procesoare și să dirijeze interogările simple către procesorul de care sunt legate discurile, purtând informația necesară interogării respective.

Obiectivele sunt creșterea performanțelor SGBD-urilor, atât pentru activitatea tranzacțională cât și pentru cea decizională în exploatarea bazelor de dimensiuni mari (de ordinul terabajților).

Integrarea tehnologiei bazelor de date cu tehnologiile multimedia

O altă tehnologie care a afectat, firesc, și domeniul bazelor de date a fost tehnologia multimedia. Conceptul tradițional de multimedia se referea la capacitatea noilor suporturi de a stoca informația de natură diversă: sunet, imagine, text, animație. Bazele de date au integrat și dezvoltat tehnologia multimedia și oferă acum utilizatorilor posibilități de *stocare* și de *manevrare* a informației de toate tipurile.

Integrarea tehnologiei bazelor de date cu programarea orientată spre obiecte

În programare, modelul orientat spre obiecte a fost propus ca model natural de structurare a datelor complexe, fiind, incontestabil, cel mai popular model din ultimii ani. La sfârșitul anilor 1980 s-a născut ideea de a importa acest model în bazele de date. Sistemele de baze de date orientate spre obiecte s-au născut din convergența a două tehnologii: pe de o parte funcționalitățile au fost importate din **tehnologia bazelor de date relaționale** (interogare declarativă, tranzacții, menținerea restricțiilor de integritate etc.), iar pe de altă parte, modelul de date a fost adoptat din **tehnologia limbajelor de programare orientate spre obiecte**.

Integrarea tehnologiei bazelor de date în sistemele inteligente

Bazele de date deductive s-au dezvoltat datorită interesului manifestat pentru logica predicatelor ca metodă de introducere a raționamentelor artificiale în sistemele informatice. Pentru că logica predicatelor este o metodă puternică de reprezentare a cunoașterii, iar mecanismul inferențial bazat pe calculul predicatelor este suficient pentru multe aplicații inteligente, implementarea acestui mecanism în sistemele clasice de gestiune a bazelor de date a condus la apariția **bazelor de date deductive**, denumite **baze de cunoștințe**.

Bazele de date inteligente reprezintă o nouă tehnologie pentru managementul informației, rezultată din integrarea tehnologiilor convenționale pentru baze de date cu cele mai recente tehnologii referitoare la programarea orientată spre obiecte, sistemele expert, sistemele hipermedia și regăsirea on-line a informației.



sarcini de laborator

- 1.** Căutați pe Internet personalități în domeniul tehnologiei informației și realizați o prezentare PowerPoint despre una dintre ele (cine este, ce contribuție și-a adus, unde lucrează, de ce credeți că acea persoană a devenit lider și nu s-a mulțumit cu un simplu job, ce calități, interese, aptitudini personale are, de ce ați ales-o).
- 2.** Scrieți un eseu „Ce știu despre mine!“ urmărind răspunsuri la întrebări de felul următor:
 - Ce îmi place să fac? Ce meserie aș face dacă aș fi plătit și ce aș face chiar gratis pentru că îmi place foarte mult?
 - Ce știu să fac? La ce sunt bun?
 - Unde mă voi duce după terminarea scolii? Ce carieră mi se pare cea mai probabilă, judecând după calitățile mele?

3. Căutați pe Internet job-uri în domeniul IT și scrieți cerințele fiecărui job (studii, calificări, trăsături de personalitate cerute). Aflați ce salariu are un administrator de baze de date. Dar un analist-programator în domeniul bazelor de date? Aflați ce alte avantaje vă oferă lucrul în domeniul IT!

4. Grupați-vă câte 3 elevi. Alegeți un domeniu de activitate (sănătate, învățământ, transport, comerț, industrie, agricultură etc.) Căutați pe Internet un site din domeniul respectiv. Descrieți informațiile pe care le prezintă site-ul și imaginați-vă cui ar servi, la baza căror decizii ar sta. Identificați informațiile calculate și forma lor de prezentare: tabele, grafice. Identificați datele pe baza cărora s-au obținut informațiile.

Prezentați colegilor rezultatul cercetării voastre.

Baze de date și sisteme de gestiune a datelor

Baza de date (BD) poate fi definită ca un *ansamblu de date* interconectate, împreună cu descrierea lor, care răspunde calităților de centralizare, coordonare, integrare și difuzare a informațiilor și care asigură satisfacerea tuturor necesităților de prelucrare ale tuturor utilizatorilor dintr-un sistem.

Sistemul de gestiune a bazei de date (SGBD) este un pachet de programe ce permite utilizatorilor să interacționeze cu o bază de date, asigurând acesteia următoarele *caracteristici*:

1. *Independența datelor* față de programul care le gestionează. Baza de date este descrisă independent de programele utilizator. Descrierea vizează tipurile de date și legăturile dintre ele și formează dicționarul datelor, memorat odată cu baza de date.
2. *Nivel redus de redundanță*. O anumită dată se află într-un singur loc.
3. *Securitatea datelor* înseamnă protecția împotriva accesului neautorizat, în vederea extragerii sau distrugerii unor date cu caracter confidențial.
4. *Integritatea datelor* înseamnă protecția împotriva defecțiunilor hardware sau software.
5. *Transparența* presupune existența unor facilități de utilizare a datelor fără ca utilizatorii să cunoască întreaga complexitate a bazei de date.
6. *Limbajele de descriere și manipulare a datelor de nivel foarte înalt* înseamnă existența unor limbaje performante de regăsire a datelor, care permit exprimarea sub forma unor conversații a unor criterii cât mai complexe de selectare a informației și indicării unor reguli cât mai generale de editare a informațiilor solicitate.
7. *Facilități multiutilizator* înseamnă că datele pot fi accesate și chiar gestionate din diferite noduri ale rețelei de calculatoare de către diferiți utilizatori.
8. *Accesibilitatea*. Gestiunea datelor organizate în baze de date a fost preocuparea multor specialiști software, ajungându-se la oferirea unor pachete software care permit gestiunea unor date foarte complexe în condiții de eficiență maximă.

Arhitectura generală a unui SGBD

Un SGBD conține o interfață între utilizatori și baza de date și asigură, în principal, crearea, actualizarea și consultarea acestora, monitorul, procesorul de consultare și diferite compilatoare pentru limbajele de descriere și de manipulare a datelor.

Monitorul bazei de date este un ansamblu de module care realizează interfața dintre datele interne conținute în bază și programele sau comenzile de prelucrare. Principalele sarcini pot fi grupate astfel:

❶ **Interacțiunea cu gestionarul de fișiere.** Datele sunt stocate pe disc prin intermediul sistemului de gestiune a fișierelor din sistemul de operare al calculatorului pe care este instalat SGBD-ul. Monitorul SGBD asigură traducerea instrucțiunilor utilizator în instrucțiuni sistem, fiind responsabil de buna desfășurare a operațiilor de citire-scriere în/din bază.

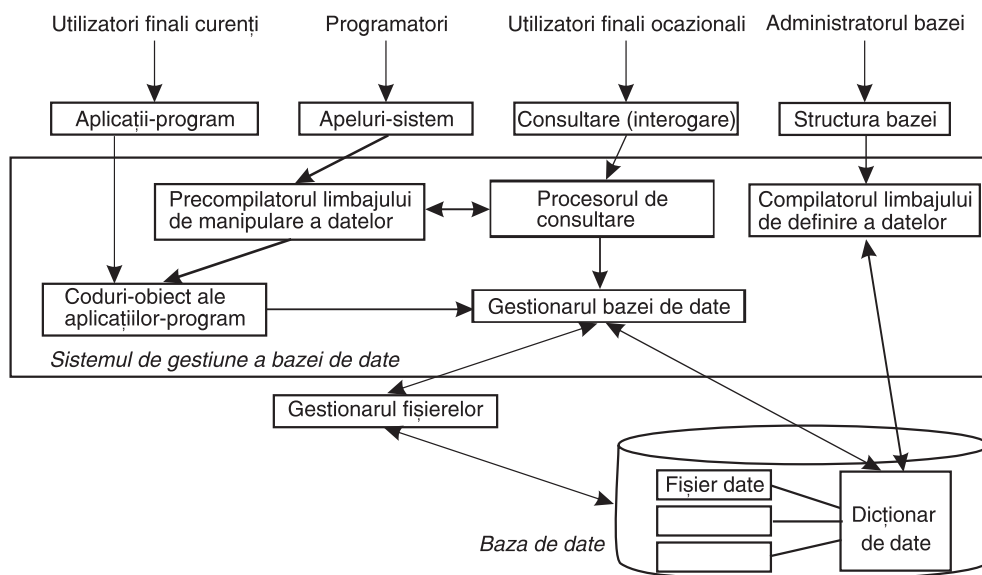


Figura 1-3: Schema de funcționare a SGBD-ului

❷ **Validarea datelor.** Datele stocate trebuie să respecte anumite restricții de integritate, specificate de administratorul bazei de date. După implementarea mecanismului de integritate, monitorul SGBD verifică dacă toate actualizările se derulează cu respectarea restricțiilor.

❸ **Securitatea datelor.** Accesul la date trebuie să fie selectiv și să verifice drepturile de acces ale fiecărui utilizator.

❹ **Salvare și restaurare.** Accidentele ca întreruperile de curent electric, erorile de programare, distrugerea suprafeței magnetice pot distruge coerența bazei de date pentru că datele care erau în curs de prelucrare se pot pierde sau altera. Monitorul are dificilă sarcină de a detecta la timp avariile și de a restaura datele pierdute în forma și conținutul de dinaintea accidentului, prin intermediul unor proceduri speciale de salvare și restaurare.

❺ **Prelucrări concurente.** La o bază de date pot lucra simultan mai mulți utilizatori. Pentru asigurarea coerenței datelor, monitorul supervizează accesul și acordă niveluri de prioritate pentru operații.

Niveluri de structurare a datelor într-o bază de date

Obiectivul principal al unei BD este de a separa descrierea datelor și programele de aplicații. Pentru a realiza acest obiectiv este necesară o abstractizare a datelor memorate în BD; astfel s-a ajuns la separarea a 3 niveluri de organizare și percepție a BD: EXTERN, CONCEPTUAL și INTERN. La fiecare nivel se definește o schemă (model) a bazei de date.

a. Structura virtuală sau **structura globală (externă)** înseamnă toate datele necesare tuturor utilizatorilor unei baze de date, în condiții de redundanță minimă și controlată a datelor. Descrierea acestui nivel (ansamblu de date, legături, proprietăți) se face într-un limbaj de descriere special al SGBD, numit SCHEMĂ CONCEPTUALĂ. Schema este unică, este memorată pe suportul fizic și este invocată la fiecare solicitare a unui subset de date de către un program-aplicație. Ea este realizată de administratorul bazei de date.

b. Structura logică sau **structura conceptuală** se referă la forma în care fiecare utilizator vede datele, în funcție de aplicația pe care încearcă să o rezolve. Descrierea structurii logice a datelor se numește SUBSCHEMĂ și este un subset de date necesare unui program.

c. Structura fizică sau **structura internă** este nivelul elementar la care se pot constitui datele; se referă la modul de memorare a datelor pe suportul de stocare. Acest lucru este interesant pentru un inginer de sistem sau un programator la nivelul limbajului de asamblare. Fișierele cu date sunt stocate în memoria externă în diferite structuri interne, care permit utilizarea eficientă a suportului și minimizarea timpului de acces. La acest nivel se numește SCHEMĂ INTERNĂ.

Terminologie și concepte specifice bazelor de date

Elementul fundamental al modelului conceptual este **entitatea**, ca termen generic ce desemnează obiectele similare ca structură, dar care sunt identificabile, deci se pot deosebi între ele prin trăsături specifice. O entitate este formată dintr-o mulțime de **atribute (caracteristici)** care pot defini complet obiectul. Lista atributelor care pot caracteriza o entitate se numește **familie de caracteristici** sau **structură conceptuală**.

Valorile familiei de caracteristici corespunzătoare unui obiect distinct poartă numele de **realizare de entitate** sau instanță. Caracteristica asociază câte o valoare dintr-un domeniu fiecărei realizări a respectivei entități. **Domeniile** pot fi deci numere întregi, șiruri etc. Un atribut sau un set de atribute care identifică în mod unic fiecare realizare a unei entități se numește **cheie**. Instanțele sunt distincte.

Relațiile dintre două entități ale unei baze de date pot fi:

1. Relația 1-1 (unu-la-unu) – de exemplu, între entitatea Clase și Diriginți: o clasă are un singur diriginte, un diriginte are o singură clasă;
2. Relația 1-n (unu-la-mulți) – de exemplu, între entitatea Clase și Elevi: o clasă are mai mulți elevi, un elev aparține unei singure clase;
3. Relația n-n (mulți-la-mulți) – de exemplu, între entitatea Clase și Profesori: o clasă are mai mulți profesori, un profesor predă la mai multe clase.

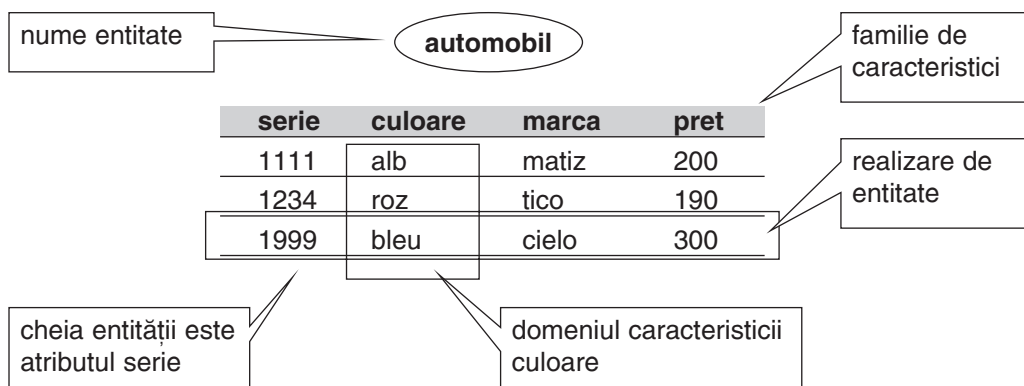


Figura 1-4: Exemplu de folosire a terminologiei bazelor de date

O entitate poate fi definită prin liste diferite de caracteristici, în funcție de sistemul informațional căruia îi aparține.¹

Observație. Convenim să folosim următoarele simboluri în reprezentarea schemei conceptuale: elipsa pentru entitate, săgeata dublă pentru relația N-N, săgeata simplă pentru relația 1-N și o linie pentru relația 1-1.

Modele conceptuale ale bazelor de date

Analiza, proiectarea și implementarea structurii conceptuale a bazei de date sunt realizate utilizând un **model de date**. Un asemenea model reprezintă un ansamblu de instrumente conceptuale care permit descrierea datelor, a relațiilor dintre ele și a restricțiilor la care sunt supuse.

Dacă reprezentarea într-o bază de date a entităților este relativ simplă, iar structura de tip fișier ar fi suficientă, modul de memorare a asocierilor constituie piatra de încercare a eficienței implementării unei aplicații de baze de date, deci cheia de boltă a unui model conceptual.

a. Modelul ierarhic

Este considerat primul model utilizat în structurarea bazelor de date și presupune ierarhizarea entităților într-o structură de tip arbore,

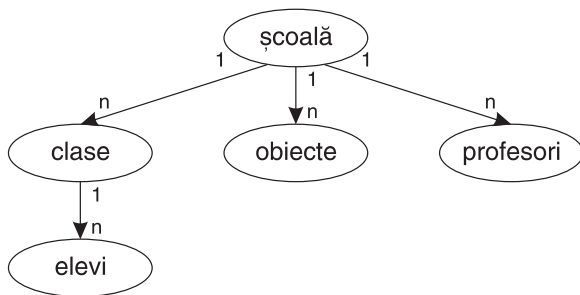


Figura 1-5: Exemplu de model ierarhic

¹ entitatea AUTOMOBIL ce desemnează un obiect în cadrul sistemului informațional al unei fabrici de mașini poate fi descrisă prin atributele *tip-motor*, *model*, *număr-de-portiere*, *forma-caroseriei* etc. Dacă însă entitatea AUTOMOBIL se referă la obiectele supuse vânzării într-o unitate comercială, atunci va trebui să conțină: *seria-motorului*, *firma producătoare*, *prețul* etc.

fiind acceptate doar relații tip 1-1 sau 1-n (ȘCOLI→1, n→PROFESORI). Dacă însă un profesor predă la mai multe școli, atunci informațiile personale vor fi repetate în entitatea PROFESORI (vorbim de o anumită redundanță a datelor); de asemenea, modelul are dificultăți atunci când sunt necesare includeri sau excluderi de entități sau de relații².

b. Modelul rețea

Este un model mai general de organizare a datelor, în care nu mai există restricția ca un nod să nu aibă decât un singur ascendent. Pot fi modelate și relațiile n-n (ȘCOLI→n, n→PROFESORI).

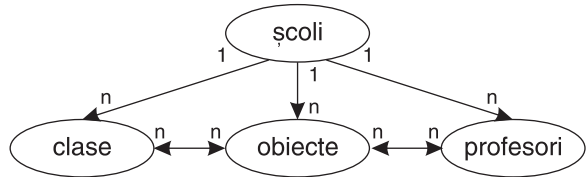


Figura 1-6: Exemplu de model rețea

c. Modelul relațional

În bazele de date relaționale, entitățile sunt organizate în tabele simple, bidimensionale, fără legături fixe. Relațiile necesare sunt stabilite prin asocierea între ele a unor câmpuri cheie ale fiecărei tabele.

De exemplu, entitățile CLASE și ELEVI sunt în legare prin câmpul cod-clasă. Legătura este de tip 1-N.

CLASE		ELEVI			
cls	diriginte	cod_cls	nume	media	absente
9a	Popa Ion	9a	Enache	9.40	12
9b	Albu M	9a	Dumitru	8.77	10
		9b	Albu ion	9.66	0

Figura 1-7: Exemplu de model relațional

Modelul relațional este caracterizat prin unitatea și simplitatea reprezentărilor: totul se reduce la tabele. De asemenea, modelul păstrează rigoarea fundamentării sale matematice, fapt care a permis definirea unor limbaje de nivel foarte înalt (în special SQL) care utilizează elemente de algebră relațională.

d. Modelul obiect

Datele sunt organizate sub forma unui graf. Un obiect poate avea referințe la alte obiecte, în același fel în care structurile sau clasele în C++ pot avea pointeri la alte structuri sau clase. Spre deosebire însă de SQL, în limbajul de cereri orientat spre obiecte (OOQ) celelalte tipuri de acces la date se realizează utilizând un limbaj de programare procedural (C, C++, Java, SmallTalk).

e. Modelul obiect-relațional

Este o abordare recentă, care prevede extinderea modelului relațional pentru a încorpora elemente de programare orientată spre obiecte, dar cu păstrarea tehnologiilor

² Pe baza modelului ierarhic au fost concepute primele SGBD-uri, cel mai reprezentativ fiind IMS (Information Management System) al firmei IBM.

relaționale. Noul val se numește **obiect-relațional**. Modelul de date este practic același ca la modelul orientat spre obiecte. Limbajul de interogare declarativ este în esență același OQL, dar păstrează sintaxa limbajului SQL, cu extensii pentru funcțiile noi.

Cum se proiectează o bază de date?

Problema proiectării unei structuri eficiente pentru datele necesare unei probleme este cheia întregii aplicații. În general, în aplicațiile de gestiune problema cea mai mare nu o reprezintă implementarea aplicației de exploatare a bazei de date, ci proiectarea unei structuri care să permită accesul cât mai rapid la date și care să sufere de cât mai puține anomalii. După cum se spune și este demonstrat practic, dacă se dă o problemă, putem rezolva 50% din ea proiectând structurile adecvate.

Pentru modelarea unei baze de date sunt parcurse câteva faze care se încadrează în metodologia generală de dezvoltare a aplicațiilor informatice. Decupăm acum numai aspectele legate de proiectarea bazei de date.

Prima fază este **analiza** sistemului informațional existent, în care se cercetează: 1) informațiile, sursele și destinațiile lor; 2) principalele activități legate de date: prelucrări, reguli de calcul, validări, puncte de control, modalități de arhivare; 3) sistemul de codificare existent etc.

Analiza sistemului informațional existent se poate face prin două metode. Prima, cea tradițională, este *metoda procedurală*, în care accentul este pus pe fluxurile de prelucrări, întrebarea fundamentală fiind: „**ce date sunt necesare pentru a realiza sarcina X?**” Deci se pornește de la studiul cererilor de informații (informații de ieșire) și, mergând pe fluxul invers al prelucrărilor, se determină informațiile de intrare, cele care trebuie colectate, validate, organizate într-o bază de date.

O altă metodă de analiză este cea orientată spre obiecte, în care întrebările sunt de felul „**Ce clase de obiecte distincte pot fi definite pe domeniului aplicației? Cum se comportă ele? Ce evenimente pot apărea? Ce acțiuni sunt posibile?**”

A doua fază este proiectarea entităților și a relațiilor dintre ele; obținem astfel SCHEMA CONCEPTUALĂ.

Studiu de caz

BIBLIOTECA

Analiza unei probleme și proiectarea bazei de date

Ne propunem să ilustrăm schematic derularea activităților de analiză și proiectare pentru o aplicație informatică la biblioteca liceului nostru. Pentru analiza sistemului existent, culegem date și le sistematizăm, răspunzând la întrebări de felul următor:

Ce activități se desfășoară la bibliotecă? Ce decizii se iau? Pe baza căror informații? Ce date pot conduce, imediat sau după anumite prelucrări, la informațiile necesare decidenților? Cine culege aceste date? Care sunt documentele purtătoare de date? Cu ce periodicitate sunt actualizate datele? Ce colecții de date ar fi necesare? Ce structură trebuie să aibă?

1. Lista activităților are ca scop identificarea informațiilor manipulate:

Activitățile desfășurate	Periodicitate	Documente
1. Comanda și achiziționarea unor cărți	Ocazional	Oferte, comenzi, facturi
2. Înregistrarea cărților sosite în bibliotecă într-un registru inventar	La sosirea cărților	Comanda, registrul inventar
3. Inventarul cărților existente	Anual	Registrul inventar
4. Scoaterea din inventar a cărților deteriorate sau pierdute	Ocazional	Registrul inventar
5. Completarea fișelor pentru cititorii noi	La înscrierea cititorului	Fișa cititorului
6. Anularea fișelor pentru cititorii plecați sau penalizați	Ocazional	Fișa cititorului
7. Împrumuturi și restituiri	Zilnic	Fișa cititorului
8. Afișarea unor situații centralizatoare	Anual	Fișa cititorilor, registrul inventar, situație centralizatoare
9. Elaborarea fișelor de carte și aranjarea lor alfabetic pe autori, edituri, domenii etc.	La sosirea cărților	Fișa cărții
10. Penalizarea cititorilor	Întârziere, pierdere	Fișa cititorului

2. Lista purtătorilor de informații

Cod	Documente	Periodicitatea de completare
RI	Registrul inventar	Ocazional, la sosirea/scoaterea cărții din inventar
FC	Fișa cărții	Ocazional, la sosirea unei cărți
FCIT	Fișa cititorului	La înscriere, împrumut/restituire și retragere
LI	Lista de inventar	Anual; alcătuită de comisia de inventar
OFER	Lista ofertelor de carte	Ocazional; vine de la edituri
C	Comanda	Ocazional; pe baza ofertelor; se trimite la edituri
F	Factura	Ocazional; vine de la editură odată cu livrările
SIT	Situația centralizatoare	Anual

3. Grila informațională

Documente Informații	RI	FC	FCIT	Li	Of	C	F	SIT
Titlu	*	*		*	*	*	*	
Autor	*	*		*	*	*	*	
Domeniu	*	*			*			
Editura	*	*			*	*	*	
Adresa-editurii					*	*	*	
Date-pers autori		*						

Documente Informații	RI	FC	FCIT	Li	Of	C	F	SIT
Pret-carte	*				*	*	*	
Numar inventar	*	*		*				
Isbn	*	*	*	*	*	*	*	
Comision					*			
Data	*	*	*		*	*	*	*
Numar-comanda						*		
Numar-factura							*	
Cnp-cititor			*					
Date-pers cititor			*					
Operație imprumut/restituire				*				
Numar-restantieri								*
Numar carti				*	*	*	*	*

Proiectarea entităților

Edituri (nume, adresă, alte_informații); Autori (nume, date-personale, alte_informații); Carti (ISBN, titlu, cod-autor, preț, alte_informații); Oferte (editura, carte, comision, alte_informații); Comenzi (data, numar-comanda, cantitate, alte_informații); Facturi (data, numar-factura, nr-bucăți-trimise, alte_informații); Inventar (număr-de-inventar, ISBN, starea, alte_informații); Cititori (cnp, date-personale).

Proiectarea legăturilor:

1. O editură poate trimite N oferte; o ofertă nu aparține mai multor edituri.
2. O ofertă poate constitui baza a N comenzi; o comandă nu poate să facă referire la mai multe oferte.
3. O comandă poate fi onorată prin N facturi; o factură nu este trimisă decât în contul unei singure comenzi.
4. O editură poate avea N cărți; o carte nu poate aparține mai multor edituri.
5. O carte poate avea mai mulți autori; un autor poate scrie N cărți.
6. Un cititor poate împrumuta mai multe cărți; o carte poate fi citită de mai mulți cititori.

Proiectarea SCHEMEI CONCEPTUALE

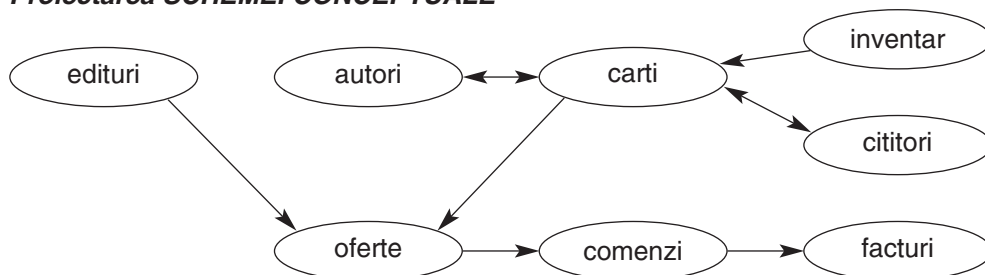


Figura 1-8: Schema conceptuală a bazei de date



sarcini de laborator

Citiți cu atenție următoarea problemă.

Pe baza dosarului candidatului, alcătuit din certificatul de naștere (CN), fișa de înscriere (FI) și avizul epidemiologic (AE), secretariatul școlii înregistrează numele, școala de unde vine elevul, profilul la care se înscrie pentru admitere în registrul de înscriere (RI) și trece pe dosar numărul de ordine.

După terminarea înscrierilor, se completează o listă alfabetică a candidaților (LA1) în două exemplare, dintre care unul este afișat la poarta școlii. Se face ordonarea alfabetică și se repartizează candidații pe săli de examen (trebuie verificate sălile și numărul de locuri din fiecare sală).

Numărul de locuri la fiecare profil al școlii este decis de forurile superioare și înscris pe un document numit plan de școlarizare (PS). El este trimis școlii înaintea începerii înscrierilor.

Profesorii din comisie cu numele, gradul didactic, școala unde sunt titulari, specialitatea etc. sunt înscrși pe documentul *lista comisiei de examen* (LC) de către inspectorat și trimis secretariatului liceului.

Pe baza listei alfabetice a candidaților (exemplarul rămas la secretariat, nu cel de la poartă) și a listei comisiei se scrie catalogul de examen (C).

După susținerea probelor, se trec notele pe catalog, se semnează catalogul de comisie și se calculează, de către secretariat, mediile fiecărui candidat. Se afișează în cel mai scurt timp lista alfabetică a candidaților (LA2), lista admișilor (LA) și lista respinșilor (LR). Aceste documente au de fapt același format, dar datele sunt diferite!

Comisia întocmește un proces verbal de examen (PV) în care va trece situațiile statistice cerute: număr înscriși, număr reușiți, cea mai mare medie la matematică etc. Catalogul, împreună cu un exemplar din lista alfabetică de după examen și cu procesul verbal (primul exemplar) se depun în arhiva școlii. Un exemplar din PV merge la inspectorat. Dosarele candidaților rămân la secretariat.

Secretariatul școlii ar dori o aplicație informatică care să simplifice munca de evidență a datelor despre candidați, comisii, săli de examen și, desigur, să permită raportarea informațiilor către decidenți în formatele și momentele solicitate.

Faceți analiza și proiectarea bazei de date necesare secretariatului liceului.

Baze de date relaționale

- *Concepte specifice*
- *Normalizarea tabelor în bazele de date relaționale*

Modelul relațional își datorează numele noțiunii matematice numită **RELAȚIE**. O relație poate fi simbolizată astfel: **R(a1, a2, a3, ..., an)**, unde R=numele relației, a1, a2, ... sunt numele atributelor sau ale constituenților. Prin definirea și utilizarea operatorilor relaționali, teoria relațiilor permite fundamentarea cercetărilor efectuate în domeniul proiectării bazelor de date relaționale și al limbajelor relaționale. Algebra relațională conține, pe lângă operatorii de mulțimi care tratează relațiile ca pe mulțimi de elemente cu operațiile cunoscute (reuniunea, intersecția, diferența, produsul cartezian) și operatorii relaționali specifici (selecția, proiecția, compunerea sau joncțiunea).

Fără a intra în fundamentele matematice ale modelului relațional, dorim să ne familiarizăm cu termenii și conceptele specifice acestui model.

O relație este o tabelă; o realizare este o linie sau un tuplu; un atribut este o coloană.

Concepte specifice bazelor de date relaționale

Observați corespondența între termenii generali acceptați de teoria bazelor de date și termenii specifici modelului relațional în figura următoare:

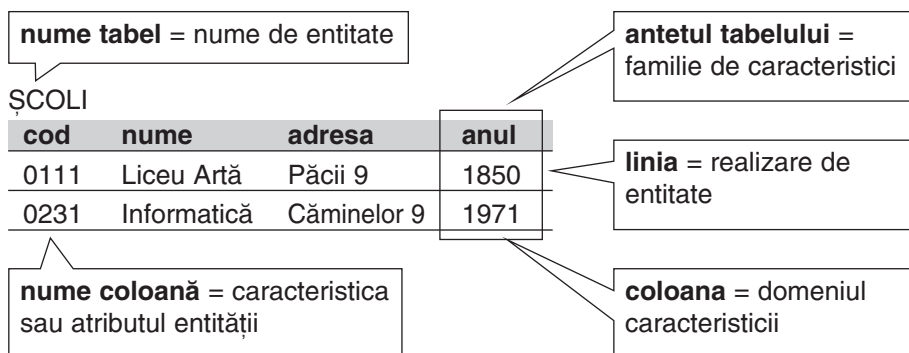


Figura 2-1: Exemplu de entitate într-o bază de date relațională

Cheia primară a unei relații este un atribut (sau grup) care identifică fără ambiguitate fiecare linie a relației. Atunci când cheia este compusă, nici un atribut al său nu poate fi eliminat fără distrugerea unicității tuplului.

O **cheie străină** este un grup de atribute care pune în legătură linia din două relații (tabele).

Legătura dintre tabele este stabilită între o tabelă (numită **părinte**) și o alta (numită **copil**) prin intermediul unui câmp comun.

Ca efect, atunci când se deplasează pointerul de fișier în tabela părinte, automat se poziționează și pointerul fișierului copil pe primul articol care are valoarea cheii egală cu cea din fișierul părinte.

Tipuri de relații

A. Relația 1-1: Un articol al tabelii părinte are legătură cu un singur articol al tabelii copil sau cu nici unul.

Exemplu

Fie entitățile Clase $\rightarrow 1, 1 \rightarrow$ Profesori cu relația „diriginte”.

Entitatea Clase are atributele cod, profil, sala, iar cod este cheia unică de identificare. Entitatea Profesori are atributele cod, nume, specialitatea, iar cod este cheia de identificare. Pe baza acestei relații putem afla pentru o clasă X care este specializarea profesorului diriginte și plecând de la profesorul X putem afla care este profilul clasei la care este diriginte, dacă are această atribuție.

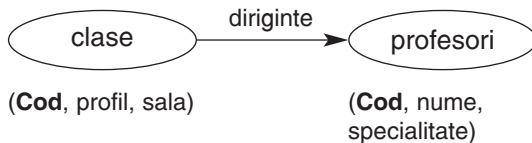


Figura 2-2: Exemplu de relație 1-1

În modelul relațional atributele vor fi grupate în două tabele:

a. tabela CLASE(cod, profil, sala, cod-dirig), unde cod este codul unic al clasei și este cheie **primară**; cod-dirig este codul profesorului diriginte și este cheie **externă** pentru că face legătura cu tabela Profesori și este un atribut unic, deoarece o clasă nu poate avea decât în singur diriginte.

CLASE			
Cod	Profil	sala	Cod-Dirig
12a	Info	P2	101
12b	Științe	15	104
12c	Litere	P6	102

b. tabela PROFESORI(cod, nume, specialitate, cls-dirig), unde cod este codul unic al profesorului **cheie primară**, clasa-dirig este codul clasei la care este diriginte și este cheie **externă unică** pentru că face legătura cu o singură linie din tabela Clase.

PROFESORI			
Cod	nume	specialitate	Clasa-Dirig
100	Popa	Mate	
101	Albu	Biologie	12a
102	Barbu	Informatica	12c
103	Enache	Chimie	
104	Anania	L.Romana	12b

B. Relația 1-n: Unui articol al tabelii părinte îi corespunde unul sau mai multe articole ale tabelii copil, iar unui articol al tabelii copil îi corespunde un singur articol în tabela părinte.

Exemplu

Fie entitățile Clase $\rightarrow 1, n \rightarrow$ Elevi, cu relația „învățată”. Entitatea Clase are atributele cod, profil, sala, iar cod este cheia de identificare unică.

Entitatea Elevi are atributele cod, nume, adresa, medie, iar cod este cheia de identificare. Pe baza acestei relații putem afla pentru o clasă X care sunt elevii clasei, iar pentru elevul X putem afla care este profilul și dirigintele clasei în care învață.



Figura 2-3: Exemplu de relație 1-n

Relația 1-n presupune crearea a două tabele în modelul relațional:

CLASE(cod, profil, sala, cod-dirig), unde cod este codul unic al clasei și este cheie **primară**;

ELEVI(cod, nume, media, cod-clasa) unde cod este codul unic al elevului și **cheie primară**, iar cod-clasa este codul clasei unde învață elevul și este cheie **externă** pentru că face legătura cu tabela Clase. Valorile acestui atribut nu sunt unice – pentru că o clasă poate avea mai mulți elevi.

Observație: În SGBDR nu pot fi reprezentate relații de tip n-n. Ele vor fi „sparte“ în procesul proiectării bazei de date în relații 1-1 și/sau relații 1-n.

CLASE			
Cod	Profil	sala	Alte-informatii
12a	Info	P2	
12b	Științe	15	
12c	Litere	P6	

ELEVI			
Cod	Nume	Media	Cod-clasa
1	Ionescu	7.50	12b
2	Albulescu	10	12c
3	Enachescu	8.90	12c
4	Enache	10	12a
5	Anania	8.00	12b

O **schemă relațională** poate fi definită ca un ansamblu de relații asociate semantic prin domeniul lor de definiție și prin anumite **restricții de integritate**. Schema relațională este independentă în timp.

Păstrarea integrității unei baze de date se realizează prin reguli de integritate, ce sunt memorate odată cu structura bazei și se verifică la orice acțiune asupra bazei. Regulile de integritate pot fi:

- Restricțiile cheilor primare – se referă la condiția de unicitate și de valori non-nule;
- Restricții referențiale – apar atunci când o tabelă este în relație cu alta. De exemplu, cheia străină nu trebuie să aibă valori care nu se regăsesc drept valori ale cheii primare în tabela de referință.
- Restricții de comportament – pot fi impuse asupra valorilor diferitelor atribute sau asupra întregii înregistrări.

Procedurile stocate sunt secvențe de program care fac parte integrantă din baza de date și sunt încărcate automat în memorie la deschiderea bazei. **Declanșatoarele (triggers)** sunt folosite pentru validarea tabelelor, a valorilor unor articole sau atribute etc. Numele de declanșatoare poate fi explicat prin faptul că procedura este invocată automat când un eveniment (adăugare, inserare, ștergere) modifică o tabelă. Principalul lor obiectiv este păstrarea integrității referențiale a bazei de date.

Tabelele virtuale sau vederile (views)

O altă noțiune a modelului relațional este *imaginea* sau *tabela virtuală* sau *vederea (view)*. O astfel de tabelă este o relație dinamică, fiind definită pe baza unei expresii relaționale dintre atributele tabelelor reale. **Ea nu păstrează conținutul tabelelor de bază, ci doar schema.** Tabelele virtuale oferă posibilitatea prezentării numai anumitor date (securitatea față de anumiți utilizatori). Odată definită, tabela virtuală este memorată în baza de date și poate fi actualizată în sensul actualizării simultane a tuturor tabelelor de bază din care este alcătuită.

De exemplu, definim o tabelă virtuală pe tabelele Elevi și Clase, care reține doar numele elevilor și profilul clasei în care învață.

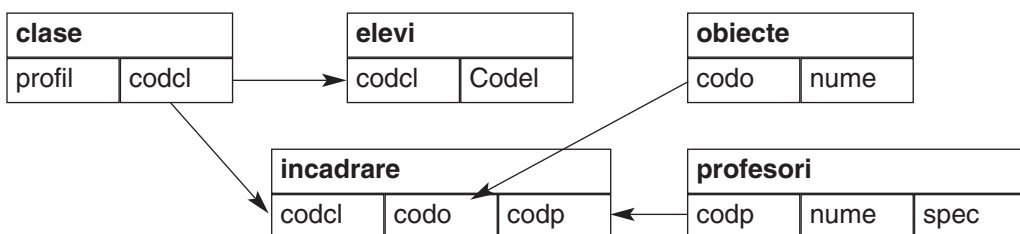
ELEVI		CLASE		PROFIL_ELEVI	
nume	Codcl	codcl	Profil	nume	Profil
Albu	9a	9a	Info	Albu	Info
Barbu	9c	9b	Mate	Barbu	Bio
		9c	Bio		

Orice modificare a datelor din tabelele sursă se reflectă automat în tabela virtuală.

Studiu de caz

Definirea restricțiilor de integritate

Definim în contextul bazei de date SCOALA următoarele tabele: CLASE, ELEVI, PROFESORI, OBIECTE, INCADRARE (codcl=codul clasei, codo=codul obiectului, codp=codul profesorului, codel=codul elevului).



Cheile primare (identificatori unici) sunt: clase.codcl; elevi.codel, Obiecte.Codo; Profesori.codp, Incadrare.(Codp+codo+codp).

Cheile străine sunt: elevi.codcl (referă tabela Clase); incadrare.codo (referă tabela Obiecte), incadrare.codcl (referă tabela Clase), incadrare.codp (referă tabela Profesori).

Restricții asupra cheilor:

- Orice valoare a atributului Codcl trebuie să identifice o linie în Clase.
- Orice valoare a atributului Codo trebuie să identifice o linie în Obiecte.
- Orice valoare a atributului Codp trebuie să identifice o linie în tabela Profesori.
- Orice valoare a grupului de atribute: codcl+codo+codp trebuie să identifice unic o linie a tablei Incadrare.

Restricții referențiale:

- La adăugarea unei linii în tabela Incadrare se va verifica dacă valoarea atributului Codp există în tabela Profesori, dacă valoarea atributului Codo există în tabela Obiecte și dacă valoarea atributului Codcl există în tabela Clase.
- La ștergerea unei linii din tabela Obiecte se vor șterge liniile cu valoarea cheii Codo egală cu valoarea cheii primare șterse.
- La ștergerea unei linii din tabela Clase se vor șterge liniile cu valoarea cheii Codcl egală cu valoarea cheii primare șterse.
- La ștergerea unei linii din tabela Profesori se vor șterge liniile cu valoarea cheii Codp egală cu valoarea cheii primare șterse.

Restricții de comportament particulare:

- La nivel de atribut: atributul Codcl va avea pe primele două poziții cifre pentru a marca anul școlar, iar al treilea caracter este o literă mică: Exemple: 10a, 9c, 12d.
- La nivel de linie: dacă codcl este „9a”-„9z” profilul aparține mulțimii „info”, „mate”, „bio”, altfel profilul este numai „info”.



sarcini de laborator

Fie următoarea schemă conceptuală a unei baze de date care reține informații despre discurile, piesele, autorii și cântăreții de romanțe și muzică populară pe plan național și internațional.

I. Proiectați SCHEMA RELACIONALĂ.

II. Verificați dacă puteți obține informațiile următoare și identificați subschemele implicate în procesul de obținere a informațiilor dorite.

1. Care sunt muzicienii autori de romanțe?
2. Care sunt formațiile americane country?
3. Câte discuri a scos vestita cântăreață Sofronia Pădureanca?
4. Cât timp durează albumul „Dă jalei și dă dor”?
5. Care sunt piesele muzicale de pe primul disc al Casei de Discuri „Trepădușul” de după Revoluție?
6. Câți membri fac parte din formația „Fluierașul”?
7. Care sunt romanțele mai lungi de o jumătate oră?

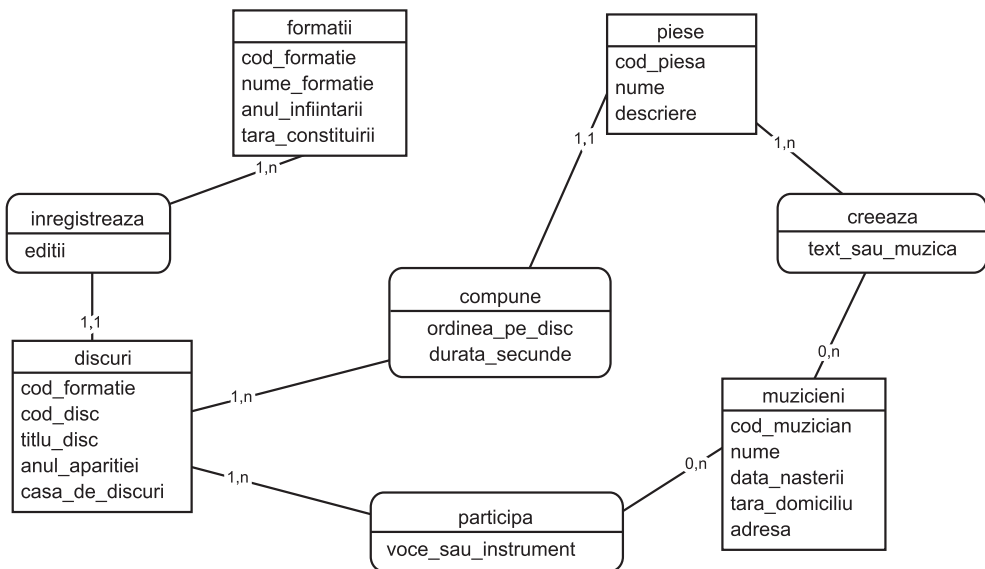


Figura 2-4: Schema conceptuală a bazei de date

Proiectarea bazelor de date relaționale: forme normale

Proiectarea unei baze de date relaționale începe prin definirea entităților și a relațiilor dintre ele. Apoi se definesc tabelele care vor memora atât datele din entități, cât și relațiile dintre acestea. Atenția proiectanților trebuie îndreptată către construirea unor tabele care să evite anomaliile de actualizare și dificultățile de prelucrare.

O tabelă de date este corect formulată – deci este o relație conform restricțiilor impuse de teoria relațiilor a lui A.F. Codd – dacă:

1. în cadrul unei baze de date are nume distinct;
2. fiecare celulă a relației conține o singură valoare;
3. fiecare atribut are un nume distinct;
4. orice valoare a unui atribut face parte din domeniul pe care a fost definit acesta;
5. ordinea dispunerii atributelor în relație nu prezintă importanță;
6. orice linie este distinctă de celelalte;
7. ordinea liniilor nu influențează conținutul informațional al relației.

Exemplu

Să presupunem că tabelul următor reține codurile produselor solicitate la o comandă, precum și numărul, data și valoarea comenzii:

com	data	furn	adr	cod1	cod2	cod3	cod4	cant	val
006	01.03.98	f1	Bc	a23	b66	c33		10	12980
007	01.09.98	f1	Bc	c33				12	12000

Verificăm respectarea definiției anterioare: nu sunt linii identice, nu sunt valori de tipuri diferite în coloane, cheia este atributul Com (număr comandă). **Dar:**

- a) *dacă există comenzi cu mai mult de 4 produse?*
- b) *dacă sunt comandate cantități diferite din toate produsele?*
- c) *unde este plasat prețul fiecărui produs?*
- d) *prelucrări de tipul «valoarea totală a comenzilor pentru produsul 'b66'» necesită verificarea tuturor coloanelor și însumarea rezultatelor pentru întreaga bază de date, lucru care conduce la un timp mare de răspuns.*

Deci se impune eliminarea câmpurilor care se repetă și astfel se ajunge la *prima formă normală*:

comanda	data	furn	adr	codprod	cant	pret	valoare
006	01.03.98	f1	Bc	a23	10	100	12980
006	01.03.98	f1	Bc	b66	10	200	12980
006	01.03.98	f1	Bc	c33	10	150	12980
007	01.09.98	f1	Bc	c33	12	150	12000

În această tabelă, cheia este compusă din atributele (comanda + codprod).

Prima formă normală este identificată cu noțiunea de relație

Dar nu toate relațiile sunt egal acceptabile, deoarece anumite relații pot conduce la dificultăți de actualizare sau de manevrare.

Operația de „spargere“ a relației care manifestă anomalii în alte relații poartă numele de normalizare.

Procesul de normalizare are la bază noțiunea de *dependență funcțională*. Dependența funcțională este o legătură între două atribute, în care al doilea poate fi determinat dacă primul este cunoscut. Exemplu: cunoscând denumirea unui furnizor putem să-i aflăm adresa. Spunem că atributul *adr* este dependent funcțional de atributul *furn* și vom nota $(furn) \gg (adr)$.

Să discutăm însă dependențele funcționale legate de cheia compusă a relației. Între (numărul comenzii + codprodus) și valoare există o dependență funcțională. Dându-se o comandă și un produs, putem găsi valoarea. Dar întrebarea este a cui valoare o aflăm? A comenzii întregi sau a produsului pe respectiva comandă? Dacă este valoarea comenzii, observăm că atributul depinde numai de o parte a cheii $(comanda) \gg (valoare)$.

O relație se află în a doua formă normală dacă toate atributele non-cheie sunt dependente de întreaga cheie.

Deci *problema apare când cheia este compusă din mai multe atribute*. O relație care are chei simple este în a doua formă normală.

Observăm că atributele non-cheie nu sunt dependente de întreaga cheie. Normalizăm și împărțim tabela în două alte tabele.

COMENZI					PRODUSE			
comanda	data	furn	adr	valoare	comanda	codprod	cant	pret
006	01.03.98	f1	Bc	12980	006	a23	10	100
007	01.09.98	f2	Gl	12000	006	b66	10	200
					006	c33	10	150
					007	c33	12	150

Să urmărim exemplul în continuare. Cheia relației la tabela COMENZI este comanda. Știind numărul de comandă, putem afla numele furnizorului. Deci $(comanda) \gg (furn)$. Știind furnizorul, putem determina adresa $(furn) \gg (adr)$. Pare că adresa depinde funcțional prin tranzitivitate de comandă, ceea ce nu este adevărat.

O relație se află în a treia formă normală dacă se află în forma a doua și nu prezintă dependențe tranzitive.

Vom normaliza în continuare și obținem alte două tabele:

FURNIZORI		COMENZI			
codfurn	adr	comanda	data	furn	val
f1	Bc	006	01.03.98	f1	12980
f2	Gl	007	01.09.98	f2	12000

Să recapitulăm:

Prima formă normală este identificată cu definiția unei relații.

A doua formă normală impune ca toate atributele non-cheie să fie dependente de întreaga cheie.

A treia formă normală presupune inexistența dependențelor tranzitive.

În concluzie, prin normalizare spectrul anomaliilor de care suferă o relație se restrânge foarte mult. Toate aceste forme normale sunt folositoare, dar niciuna nu garantează că au fost eliminate toate anomaliile!

Pentru proiectul BIBLIOTECA au fost create următoarele **variante** de structură a bazei de date, legate de desfășurarea operațiilor curente de împrumut și restituire a cărților. Analizați cerințele de actualizare a datelor și observați dacă există anomalii.

Structura bazei de date

Ce operații sunt executate?

Varianta 1

LECTURI

(nr_inventar, titlu_carte, autor, anul_intrarii, numele_cititorului, adresa, telefon, data-imprumutului)

La sosirea unei cărți adaug un articol și trec anul intrării. La scoaterea din inventar caut în tabel cartea și o șterg. La împrumutarea unei cărți completez numele cititorului, adresa, telefonul și data împrumutului. La restituirea unei cărți caut cartea și șterg doar informațiile despre cititor.

Observații

Baza de date permite desfășurarea operațiilor de actualizare a cărților, dar prezintă anomalii legate de actualizarea cititorilor.

Astfel:

- *dacă ștergem articolul corespunzător unei cărți la scoaterea sa din inventar, atunci vor fi șterse și informațiile despre cititor! În acest caz, spunem că există **anomalii de ștergere**.*
- *nu putem adăuga un nou cititor decât atunci când acesta împrumută o carte. Spunem că există **anomalii de inserare**.*
- *datele despre un cititor le găsim în toate articolele corespunzătoare cărților împrumutate de acesta și nerestituite. Spunem că există **date redundante**.*

Deficiențele se datorează reținerii într-o singură tabelă și a informațiilor despre cărți și a celor despre cititori.

Varianta 2

CARTI

(nr_inventar, titlu, autor, editura, an_intrare, an_iesire)

CITITORI

(cod_cit, nume, adresa, telefon, nr_inventar, data_imprumutului)

La sosirea unei cărți adaug o linie în Carti. La scoaterea din inventar trec anul ieșirii cărții. La împrumut caut cartea și completez în tabela Cititori codul cărții și data împrumutului. La restituire caut cititorul și șterg codul cărții și data împrumutului.

Observații

Structura bazei de date separă datele în două tabele (cele despre cărți și cele despre cititori) și permite desfășurarea operațiilor de intrare/ieșire a cărților și cititorilor. Dar există restricția împrumutului unei singure cărți, ceea ce nu este deloc credibil. Dacă tot am mers la bibliotecă... Desigur, putem fixa de la început numărul maxim de cărți pe care l-ar putea împrumuta un cititor (să zicem 3!) și structura tabelii CITITORI ar fi forma (cod_cit, nume, date-personale, cod_carte1, data1, cod_carte2, data2,

codcarte3, data3). În această situație, căutarea devine mai dificilă, atât la restituirea unei cărți (unde valoarea corespunde cu numărul de inventar a cărții restituite), cât și la împrumut (unde este loc liber). În plus, utilizarea memoriei este ineficientă – nu toți cititorii au exact 3 cărți împrumutate! Observați o astfel de tabelă! Pe de altă parte, ce ne-am face cu o bibliotecă unde este permis împrumutul a 20 cărți?

cod_cit	nume	nr1	data1	nr2	data2	nr3	data3
1	albu	1	01.09.2003	165	12.09.2002		
2	barbu			5	01.01.2002	12.12.2002	
3	carp						
4	doltu	10	01.03.2003	123	01.03.2003	12345	01.03.2002

Figura 2-5: Exemplu de tabelă

Varianta 3

CARTI

(nr_inventar, titlu, autor, editura, an_intr, an_ies, cod_cititor, data_imprumut)

CITITORI

(cod, nume, date-pers)

La sosirea unei cărți adaug o linie în Carti.

La scoaterea din inventar trec anul ieșirii cărții.

La împrumut completez în tabela Carti codul cititorului și data împrumutului.

La restituire caut cartea și șterg valoarea din codul cărții și data împrumutului.

Baza de date separă informațiile despre cărți și cele despre cititori. La împrumut se trece codul cititorului și data, iar la restituirea cărții se șterg valorile din câmpul Cod cititor și data împrumutului. Deci putem identifica cărțile nerestituite prin valoarea diferită de Null în coloana Carti. Dar este ineficient să parcurgem în bibliotecă fiecare carte pentru a afișa restanțierii!

Un alt dezavantaj ar fi legat de imposibilitatea obținerii unor situații de felul „care sunt preferințele cititorului X?” sau „de câte ori a fost citită cartea Y?” care ar necesita un istoric al împrumuturilor.

Varianta 4

CARTI

(nr-inv, titlu, autor, editura, an_intr, an_ies)

CITITORI

(cod_cit, nume, date_pers)

OPERATII

(nr_inv, cod_cit, data_imprumut, data_restituire)

La sosirea unei cărți adaug o linie în Carti.

La scoaterea din inventar trec anul ieșirii cărții.

La împrumut completez în operații un nou articol cu numărul de inventar codul cititorului și data împrumutului.

La restituire caut în operații și completez data-restituirii.

La sosirea unui nou cititor adaug o linie în Cititori.

Baza de date separă informațiile relativ constante despre cărți și despre cititori și cele curente, legate de împrumut/restituire. Rațiunea acestei separări vine din periodicitatea diferită de actualizare a datelor din cele 3 colecții. Observăm reținerea în tabela Operații doar a codului cărții și a codului cititorului pentru a putea face referire la informațiile „lungi” (detaliat) despre cartea sau cititorul care a împrumutat-o!

Se pot obține toate situațiile legate de activitatea bibliotecii, de exemplu:

- a. Pentru a vedea dacă există cititori restanțieri, vom parcurge tabela Operații și, în cazul valorilor nule din coloana data-restituirii, vom calcula numărul de zile de la împrumut.
- b. Pentru a afla dacă o carte este pe raft, este suficientă o căutare a codului cărții în tabela Operații.
- c. Numărul de cărți împrumutate în intervalul D1-D2 se poate calcula ca fiind numărul de linii din tabela operații, după un filtru care încadrează data împrumutului între D1 și D2.
- d. Pentru o interogare de felul „ce cărți preferă cititorul X” vom cerceta tot tabela Operații, filtrată pentru cititorul dorit. Vom afla numărul de cărți împrumutate pentru fiecare domeniu (sau colecție sau editura!) și vom alege maximum.

Varianta 5

CARTI

(nr_inv, date-despre_carte)

CITITORI

(cod_cit, date_pers)

IMPRUMUTURI

(nr_inv, cod_cit, data_imprumut)

RESTITUIRI

(cod-carte, cod-cititor,
data_imprumut, data-restituire)

Operațiile de împrumut sunt înregistrate în tabela Imprumuturi odată cu efectuarea lor. La restituire, caut în operații și mut articolul în tabela Restituiri.

Este realizată separarea informațiilor vechi (cele de care nu mai avem nevoie decât accidental, pentru situațiile de felul: cărțile preferate de o persoană, numărul cititorilor unei cărți, activitatea bibliotecii că număr de operații într-un interval etc.) de cele recente (cărți aflate la cititori).

Faptul că tabela Operații reține doar cărțile împrumutate și nerestituite scurtează dimensiunea tabelii și, implicit, timpul de accesare. Pe baza ei poate fi căutată rapid o carte pentru a vedea la cine este sau a afla restanțierii.



sarcini de laborator

1. Analizați și normalizați tabela ELEVI. Presupunem codul elevului ca fiind chiar numărul curent în clasă.

Clasa	Cod-elev	nume_elev	adresa	diriginte
11a	1	popa	iasi	albu
11a	2	iancu	bacau	albu

2. Să presupunem că elevii se pot specializa în diferite domenii și pot desfășura activități sportive. Să discutăm tabela următoare.

elev	specializari	sporturi
Popa	muzica	atletism
Popa	muzica	tenis
Lucy	foto	atletism
Lucy	film	tenis

Un elev poate avea mai multe specializări și poate practica mai multe sporturi. Cheia relației poate fi doar atributul Student.

3. Fie tabela următoare, conținând numele cursurilor, profesorii, numărul de ore pentru aceste cursuri, precum și numele elevilor care au optat pentru cursuri.

Nume_curs	nume_prof	nume_elev	Nr-ore	grad
Astronomie	popa	albu	100	prof.def
Astronomie	popa	barbu	100	prof.gr.1
Meteorologie	florea	doltu	200	lector

4. Normalizați următoarea relație cu situația închirierii sălilor dintr-un complex sportiv de către diferiți clienți.

client	nume_sala	taxa
popa	biliard	1200
Ion	minigolf	2000
gaby	tenis	3000
popa	cafenea	1200

Un client poate închiria mai multe săli; o sală poate fi închiriată de mai mulți clienți.

5. Evidența cazărilor turiștilor în hotelurile unei stațiuni este ținută în tabela TURIST. Este corectă evidența?

Turist	bi	hotel	taxa/loc
apopei eduard	as123456	lido	1200
dorneanu mihai	gf345675	lido	1200
enache sorin	bv512345	parc	1400

6. Fie un tabel care evidențiază situația stocurilor pe depozite. Normalizați-l!

depozit	material	furnizor	stoc
d1	suruburi	Astra	10
d1	suruburi	Zalau	20
d2	carcase	Metalcar	19

Un depozit poate avea mai multe materiale. Un material poate sosi de la mai mulți furnizori, dar un furnizor este specializat pe un anumit material.

Introducere în mediul FoxPro

- *Prezentarea generală a produsului FoxPro*
 - *Interfața*
 - *Configurarea*
 - *Obținerea asistenței*
 - *Moduri de lucru*
- *Gestiunea fișierelor*

FoxPro: prezentare generală

Originea limbajului Xbase – părintele limbajului FoxPro – este legată de un limbaj brevetat la sfârșitul anilor 1970 și portat pe sistemul de operare CP/M de către un angajat, Wayne Ratliff, care l-a denumit *Vulcan*. Sesizând posibilitățile comerciale ale unui SGBD (sistem de gestiune a bazelor de date) pe microcalculatoare, Ratliff a vândut produsul corporației Ashton-Tate, sub numele *dBASE II*, având un succes imediat și cucerind realmente piața utilizatorilor de microcalculatoare. A urmat portarea produsului pe calculatoarele IBM-PC și apariția variantei *dBASE III*, apoi a variantei *III Plus*, cu care compania Ashton-Tate domina la sfârșitul anilor 1980 piața bazelor de date pentru microcalculatoare. *dBASE III Plus a devenit standardul de facto, fiind urmat cu loialitate de către proiectanții de aplicații.*

În paralel, alte companii oferă compilatoare și sisteme de dezvoltare de aplicații *compatibile cu dBASE*, dar **optimizate** conform nevoilor utilizatorilor.

Una dintre aceste companii „pirat“ a fost Fox Software din Perrysburg, Ohio, ale cărei produse *FoxBase* și *FoxBase Plus* și-au câpătat rapid o reputație bazată pe un plus de viteză și maximă compatibilitate cu limbajul *dBASE*. Alți producători, cum ar fi Nantucket (compilatorul *Clipper*) și WordTech (compilatorul *QuickSilver*) au pus accentul pe crearea programelor executabile pure. **Până la sfârșitul anilor 1980, termenul Xbase a fost folosit pentru a descrie generic toate produsele compatibile dBASE.** Acestea aveau implementat un nucleu funcțional mai mult sau mai puțin echivalent cu limbajul și setul caracteristicilor *dBASE III Plus*, dar fiecare producător și-a urmat propria cale. Fox Software lansează în paralel cu firma Ashton-Tate produsul *FoxPro* compatibil cu *dBASE IV*, dar mult mai rapid și cu o interfață mai prietenoasă, orientată către ferestre.

Pretenția companiei Ashton-Tate de proprietate asupra limbajului *dBASE* s-a manifestat prea târziu, într-un mod care nu a făcut decât rău propriului produs și publicitate gratuită noului-venit. Procesul public intentat companiei Fox Software pentru furtul limbajului a orientat, din păcate, opinia publică în defavoarea firmei Ashton-Tate, considerând procesul ca o dovadă de slăbiciune a produsului *dBASE* în competiția cu *FoxPro*.

Ulterior, Ashton-Tate a fost cumpărată de firma Borland, care a creat produsul dBASE V, cu variante pentru sistemele DOS și Windows, iar firma Fox Software a fuzionat cu Microsoft și a dezvoltat variantele FoxPro pentru Windows, DOS, UNIX, MacIntosh etc.

FoxPro este un SGBD (sistem de gestiune a bazelor de date) care pune la dispoziția utilizatorilor aplicații complexe pentru crearea și manipularea bazelor de date, precum și pentru obținerea rapoartelor din aceste date. Dispune de un limbaj de programare propriu, limbaj procedural foarte puternic și flexibil, prin care programatorii își pot descrie datele și aplicațiile. De asemenea, are implementat limbajul de cereri SQL pentru utilizatorii neinformaticieni. Începând cu varianta 3.0, limbajul procedural FoxPro a devenit un limbaj orientat pe obiecte. Având implementat conceptul de colecție de date, el reprezintă un depozit central pentru stocarea informațiilor despre tabele.

De asemenea, FoxPro pune la dispoziția programatorilor nu numai un compilator și un mecanism performant de accesare a datelor (tehnologia Rushmore), ci și un set de utilitare puternice de proiectare, încorporate într-un mediu integrat și omogen. Acest mediu este foarte confortabil pentru proiectanții de aplicații.

FoxPro este un produs care poate rula pe platforme DOS, Windows, Unix, MacIntosh. Deși interfața și structura meniurilor sunt oarecum diferite, un programator poate trece ușor de la o variantă la alta.

FoxPro recunoaște și se adaptează automat la mediile multiutilizator, fără a fi nevoie de o variantă specială pentru rețea.

FoxPro permite comunicarea cu alte aplicații (de exemplu, Excel) prin mecanisme DDE (Dynamic Data Exchange – transfer dinamic de date). FoxPro permite schimbul de date între tabelele sale și alte aplicații în calitate de server sau de client, respectiv transmite sau primește informații către și de la programele care rulează sub Windows.

FoxPro are facilitatea OLE (Object Linking and Embedding – legarea și încorporarea obiectelor). Limbajul suportă legarea obiectelor și încorporarea lor în aplicații proprii FoxPro, cum sunt sunetele, imaginile, foile de calcul etc., create în alte aplicații Windows. În acest caz, vorbim despre FoxPro ca despre un client.

FoxPro importă și exportă date în alte formate (fișiere Microsoft Excel, dBASE, Access, Oracle, Paradox) pe diferite suporturi, local sau la distanță.

FoxPro a fost conceput în vederea unei depline compatibilități, atât cu versiunile sale anterioare, cât și cu alte produse xBase.

Modulul FoxPro Distribution Kit permite realizarea dischetelor de distribuție pentru aplicațiile executabile.

Variantele de după 3.0 permit programarea vizuală a aplicațiilor folosind generatoare – programe încadrate în generația a patra (4GL) care permit proiectarea interactivă a obiectelor cu care lucrează o aplicație.

Interfața produsului Visual FoxPro

Visual FoxPro este o aplicație Windows, deci folosește elementele de interfață specifice acestui mediu: ferestrele, meniurile, butoanele, obiectele de control. La intrarea în mediul Visual FoxPro este afișată **ferestra sistem** cu meniul principal și o fereastră de comenzi (figura 3-1).

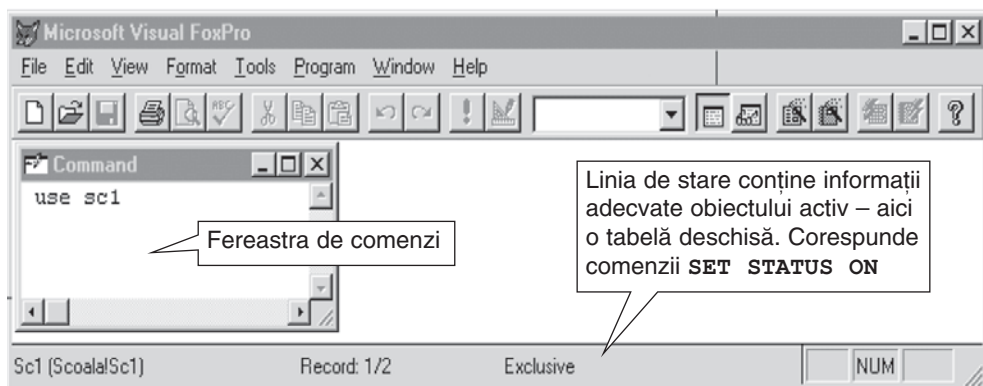


Figura 3-1: Fereastra sistem a mediului Visual FoxPro

■ **Meniul principal sau meniul Sistem** permite: operații cu fișiere (creare, deschidere, salvare, afișare la imprimantă (FILE)); operații legate de editare (EDIT); operații cu baze de date (DATABASES/TABLE); operații relative la programe (PROGRAM); utilizarea ferestrelor (WINDOW); obținerea informațiilor de asistență, alte utilitare (HELP).

■ **Meniul contextual** apare prin clic dreapta pe un item (text, icon, bară de instrumente) și conține comenzi referitoare la itemul respectiv (figura 3-2).

■ **Ferestrele standard FoxPro** sunt ferestre Windows cu aceleași caracteristici. Tipurile de ferestre sunt:

- Fereastra de comenzi**, care permite introducerea directă a comenzilor;
- Fereastra de proiectare**, care permite editarea programelor, a machetelor, a rapoartelor etc.;
- Fereastra de dialog**, care permite completarea unei acțiuni conform preferințelor utilizatorului; ca exemplu, observați fereastra Open, prin care precizați discul, directorul și fișierul care va fi deschis;
- Fereastra de mesaje**, prin care sistemul avertizează sau informează asupra efectelor unei acțiuni (erorare, riscante) a utilizatorului, solicitând confirmarea privind modalitatea de continuare sau abandonarea acțiunii.

■ **Obiectele de control** sunt alte elemente de interfață cu funcții și aspect recunoscut din mediul Windows. Lucrul cu obiectele de control este cunoscut din lecțiile anterioare.

Exemple

- Casete de editare – Edit box
- Butoane de comandă – Command button
- Liste și casete combinate – List și Combo box
- Contoare – Spinner
- Comutatoare sau casete de validare – Check box



Figura 3-2: Exemplu de meniul contextual

Configurarea mediului FoxPro

La instalare, sistemul se **autoconfigurează** la parametrii impliciți, considerați optimi de marea majoritate a utilizatorilor săi.

Pentru a îmbunătăți performanțele de lucru, multe setări pot fi redefinite pe parcurs, în funcție de particularitățile hardware ale calculatorului și de opțiunile utilizatorilor. Setările pot fi temporare (anulate la ieșirea din Fox) sau permanente (memorate în fișierele speciale de configurare).

Configurarea se poate face prin scrierea valorilor dorite a comenzilor **SET** (de configurare) direct în fereastra de comenzi sau prin folosirea ferestrei Options (figura 3-3), deschisă din meniul principal, selectând **Tools, Options**. Fereastra Options are mai multe secțiuni (tab-uri) care conțin informații diferite de configurare.

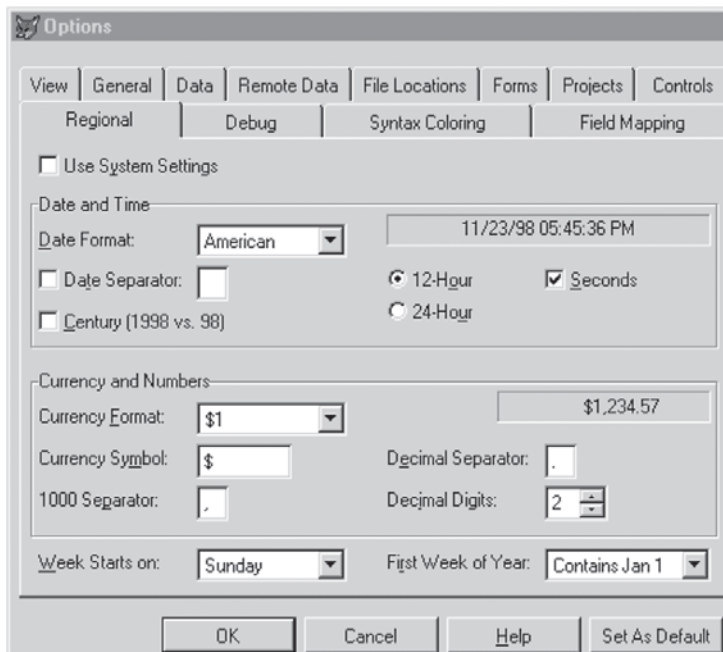


Figura 3-3:
Fereastra Options

Moduri de lucru

Spre deosebire de limbajele de programare algoritmice pe care le-am studiat în anii anteriori (PASCAL, C), în care, la rezolvarea unei probleme, oricât de simple, trebuia să scriem un program, să-l compilăm și să-l lansăm în execuție, pachetul de programe FoxPro ne permite o varietate de modalități de lucru. Acestea vor fi prezentate pe scurt în continuare.

Modul de lucru comandă

Este o modalitate de lucru interpretativă, care permite ca, imediat ce introduceți o comandă, sistemul să o cerceteze și, dacă este corectă, să o execute. Instrucțiunile sunt introduse într-o fereastră de comenzi care funcționează ca un istoric al comenzilor

introduse. Comenzile introduse în fereastra de comenzi sunt comenzi linie; apăsarea tastei **Enter** lansează comanda în execuție.

Comenzile FoxPro sunt formate dintr-un verb care indică acțiunea și un număr de clauze care particularizează efectul comenzii în situația respectivă. Formatul general al unei comenzi este:

```
<verb> [<clauza1>] [<clauza2>]
```

În general, clauzele nu au o poziție fixă în sintaxa comenzii, iar verbul poate fi prescurtat la primele 4 caractere.

Exemplu

Pentru afișarea numelui și clasei elevilor care au media cel puțin 5 se poate scrie:

```
Browse fields nume, clasa for media>=5  
Browse for media>=5 fields nume,clasa
```

Modul de lucru program

SGBD-ul FoxPro reprezintă un mediu integrat de dezvoltare a programelor utilizator, oferind un editor de texte pentru scrierea textelor sursă, depanator, compilator și linkeditor. Se pot folosi ambele tehnici de programare: atât programarea clasică, structurată și modulară, cât și programarea orientată spre obiecte. Programele sunt văzute ca fișiere de comenzi (* .PRG).

Apelul editorului se face prin comanda `MODIFY COMMAND <fis.prg>`

Comanda deschide o fereastră de editare, folosită atât pentru crearea, cât și pentru modificarea programului sursă recunoscut prin numele `<fis.prg>`.

Lansarea programului în execuție (automat se face și compilarea) se face prin comanda `DO <fis.prg>`

Programul este executat până la terminarea fișierului sau până la întâlnirea comenzii **RETURN**, care determină revenirea în fereastra de comenzi. Dacă un fișier de comenzi este apelat din alt fișier de comenzi, spunem că este **subprogram**. Nu sunt diferențe între construirea unui program și a unui subprogram. Prin comanda **DO** se caută fișierul de comenzi indicat, se deschide, se execută liniile acestuia, se închide și se revine în programul apelant sau în fereastra de comenzi.

În programe și numai în acest context se pot folosi și **comenzile multilinie**, cum ar fi comanda **IF** din exemplul următor.

```
USE ELEVI                && deschide o tabelă  
IF nume="albu"           && test asupra numelui primei persoane  
? prenume                && comanda de afișare a prenumelui  
ENDIF                    && închide structura
```

Dacă este necesară fragmentarea comenzii, se va folosi semnul „,” (punct și virgulă). Nu există separatori între comenzi. Lungimea maximă a unei linii de comandă este de 1024 de caractere.

Comentariile sunt introduse într-un program fie prin caracterul asterisc (*) la începutul liniei, fie prin dublu ampersand (&&) în continuarea comenzii propriu-zise.

FoxPro poate să apeleze comenzi DOS prin comanda `! / RUN`

Exemplu

```
MODIFY COMMAND Salarii      && deschide ecranul de proiectare
                              Salarii.prg
DO SALARII                  && compilează și execută Salarii.prg
! DIR                        && afișează lista fișierelor din
                              directorul curent
run erase salarii.prg       && apelează comanda de
                              ștergere DOS
```

Modul de lucru asistat (interactiv sau vizual)

Acest mod presupune alegerea acțiunii dorite prin intermediul interfeței FoxPro, interfață prietenoasă, orientată spre ferestre, meniuri, obiecte de control, cunoscute în mare parte din mediul Windows. Este un mod de lucru ușor de utilizat pentru operațiile directe ale utilizatorilor asupra datelor, oferind avantajul unei viteze mari de lucru.

Pe de altă parte, proiectarea vizuală a devenit un instrument modern, cu ajutorul căruia pot realiza programe și utilizatorii neinformaticieni, fără să cunoască restricțiile sintactice ale unui limbaj de programare. Singura cerință este ca proiectantul să știe foarte bine **ce** trebuie făcut; **cum** trebuie făcut hotărăște sistemul. Instrumentele folosite în mediile vizuale sunt generatoarele (designers), asistenții (wizards) sau constructorii (builders). Pe toată perioada dezvoltării programului, utilizatorul are posibilitatea vizualizării a ceea ce va obține ca rezultat.

Abordarea vizuală a obiectelor face ca Visual FoxPro să fie considerat un mediu de dezvoltare rapidă a aplicației (RAD – Rapid Application Development). Folosind acest mediu realizăm o **proiectare vizuală** a interfeței, spre deosebire de **abordarea textuală** din mediile de programare tradiționale.



sarcini de laborator

I. Sarcini pentru familiarizarea cu elementele de interfață

Intrați în mediul FoxPro; observați meniul principal, selectați diverse opțiuni; folosind fereastra de comenzi, executați câteva operații:

- mutare
- redimensionare
- expandare
- minimizare
- închidere
- deschidere

1. Creați cu un editor de texte oarecare un fișier text denumit **APLIC.TXT**. Deschideți fereastra de editare a fișierului **APLIC.TXT**.
2. Modificați poziția ferestrei de editare; redimensionați fereastra.
3. Ștergeți vechiul conținut; scrieți un alt text. Selectați o zonă din acest text și multiplicați-o de 3 ori.
4. Selectați o zonă din acest text și ștergeți-o; duplicați textul, creând un alt fișier, denumit **APLIC2.TXT**.
5. Deschideți cele două fișiere în două ferestre, prezente simultan pe ecran.

6. Schimbați între ele pozițiile celor două ferestre; salvați și închideți ferestrele.
7. Informați-vă asupra opțiunii **CUT** din meniul **EDIT** prin utilitarul **HELP**.
8. Aflați ce face comanda **CLEAR** folosind utilitarul **HELP** contextual.
9. Capturați o zonă a ecranului într-un fișier.
10. Deschideți fișierul **APLIC2 . TXT** și încadrați titlul documentului într-un chenar cu ajutorul caracterelor speciale ASCII.

II. Exerciții pentru editarea textelor

1. Deschideți editorul de texte și realizați o cerere de înscriere a elevului Popescu Emil la cursurile de vară pentru informatică din perioada 1-30 iulie!
2. Aplicați fonturi, culori și dimensiuni diferite. Încadrați cererea într-un chenar cu linii simple.
3. Multiplicați cererea de 5 ori.
4. Înlocuiți peste tot perioada cu 1-30 august.
5. Folosiți în fiecare cerere alt nume (importantă este poziționarea pe șirul „Popescu Emil“ în vederea schimbării cu alt șir din exterior!).
6. Ștergeți peste tot cuvântul „azi“.
7. Inversați cererea 3 cu cererea 1; refaceți varianta anterioară.
8. Salvați cu numele **CERERE . TXT**; afișați pe ecran sau la imprimantă conținutul fișierului.

III. Exerciții pentru configurarea mediului FoxPro

1. Scrieți în fereastra de comenzi ? **DATE ()** . Care este efectul comenzii pe ecranul sistem?
2. Schimbați formatul datei calendaristice.
3. Completați un tabel cu formatele de dată și numele acestora:
4. Introduceți secolul.
5. Fixați alt separator între informațiile de dată: zi, luna, an.
 - a) Ce comandă este generată pentru fixarea formatului de dată?
 - b) Ce comandă este folosită pentru fixarea separatorului de dată?
6. Aflați care este semnul pentru marca zecimală!
7. Scrieți în fereastra de comenzi: ? **100/2 , 100 , 0/2**
Obțineți același rezultat? Care este numărul de zecimale?
8. Observați efectul schimbării numărului de zecimale la 5.
9. Ce efect are comanda **SET DECIMALS**? Care este numărul maxim de zecimale?
10. Ce comandă folosiți pentru fixarea numărului de zecimale?
11. Fixați simbolul monetar „lei“ plasat după valoare. De exemplu: 1250lei.

Format	nume-format
ll/zz/aa	AMERICAN
zz.ll.aa	GERMAN

Gestiunea fișierelor

La lansare, SGBD-ul FoxPro este încărcat în memoria de lucru de către sistemul de operare sub care este instalat și funcționează în concordanță cu restricțiile acestui sistem de operare. Un fișier este descris în sistemul de operare prin: nume, extensie, poziție logică pe disc (calea de directoare), dimensiune, data și ora ultimei actualizări, atributele asociate accesului la fișier, drepturi de acces. Fișierele FoxPro stochează date, programe și informații pentru generatoarele sistemului. Evidența acestui ansamblu de fișiere este dificilă, mai ales când numărul lor este mare.

Utilitarul **Project Builder** servește la gestiunea fișierelor dintr-o aplicație Fox. Îl puteți lansa selectând **File, New, Project** din meniul principal sau prin comanda **CREATE PROJECT**.

Fereastra principală a utilitarului afișează un arbore de directoare standard, construit după natura fișierelor ce pot fi gestionate:

1. **Data** conține subdirectoare pentru baze de date (.DBC), tabele izolate (.DBF), interogări (.QPR).
2. **Documents** conține subdirectoare pentru formulare (.SCX), rapoarte (.FRX), etichete (.LBX).
3. **Classes** conține bibliotecile de clase (.VCX).
4. **Code** conține fișierele de proceduri (.PRG, .FXP).
5. **Other** conține fișiere auxiliare (.BMP).

Pe baza acestui arbore cu fișierele aplicației, puteți genera o aplicație executabilă, așa cum veți vedea în lecțiile următoare. Din fereastra gestionarului de fișiere puteți realiza interactiv operațiile dorite cu fișiere: proiectarea sau crearea unui fișier, deschiderea în vederea consultării, a execuției, a modificării, ștergerea fișierului etc.

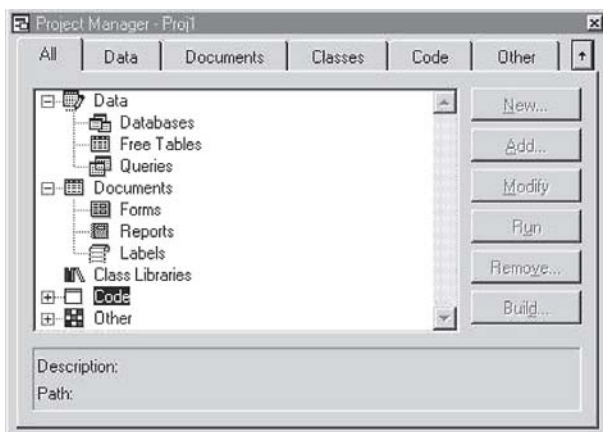


Figura 3-4: Fereastra Project Manager

Atenție

Utilitarul nu creează structura de directoare! Este doar o grupare pe care o face în funcție de tipul fișierelor.

Când începeți să lucrați la o aplicație, este bine să vă creați o structură de directoare asemănătoare celei din fereastra Project Manager; astfel veți regăsi ușor fișierele după tipul lor!

Câteva operații cu fișiere și directoare

Comandă	Efect
SET DEFAULT TO <director>	Schimbarea directorului curent.
SET PATH TO <lista dir>	Indicarea directoarelor de căutare
DIR [[ON]<disc>] [<dir>] [<sablon>] [TO PRINTER/TO FILE <fis.txt>]	Afișarea conținutului directorului curent sau a celui specificat în clauza ON. Șablonul permite limitarea listei la un grup. Clauza TO indică destinația comenzii.
COPY FILE <fis1.*> TO <fis2.*>	Copierea unui fișier identificat unic prin specificatorul de fișier <fis1.*> în alt fișier identificat prin <fis2.*>.
COPY FILE <fis1.*> TO <fis2.*>	Copierea unui fișier identificat unic prin specificatorul de fișier <fis1.*> în alt fișier identificat prin <fis2.*>.
RENAME <fis1.*> TO <fis2.*>	Redenumirea unui fișier.
ERASE <fis.*>	Ștergerea unui fișier.
=CURDIR()	Returnarea numelui directorului curent.
=FILE("<fis>")	Returnarea lui .T. dacă fișierul este găsit pe discul curent.
=GETFILE()/ =GETDIR()	Deschiderea unei ferestre pentru selectarea directă a unui fișier (director). Este returnat numele fișierului (directorului) sau șirul vid dacă apăsați tasta Cancel , Esc sau butonul Close .

Exemplu

.set default to d:\fox\exempl set path to d:\fox, c:\ use elevi	Poziționare în directorul exempl . Deschide tabela elevi . Dacă nu se găsește în directorul curent, se va căuta în D:\Fox sau c:\.
Dir dir *.* dir d:\fox*.txt	Afișează bazele de date din directorul curent Afișează toate fișierele din directorul curent Afișează lista fișierelor text din d:\fox.
set default to c:\fox	Schimbă directorul curent la c:\fox.
copy file elev.dbf to d:\fox\student.dbf	Copiază fișierul ELEV.DBF din directorul curent în directorul d:\fox sub numele STUDENT.DBF.
!copy *.txt a: Erase *.txt rename elevi.dbf to student.txt ?curdir() Cd getdir()	Copiază toate fișierele text pe dischetă. Șterge fișierele text din directorul curent. Schimbă numele fișierului. Afișează numele directorului curent. Poziționare în directorul al cărui nume îl introduceți de la tastatură.



sarcini de laborator

I. Urmăriți comenzile, verificând dacă sunt corecte sintactic și încercând să compuneți structura discului C. Atenție! Presupunem că apelul FoxPro s-a făcut prin comanda: `C:\>FOX\FOX.EXE`, deci directorul curent este rădăcina discului C.

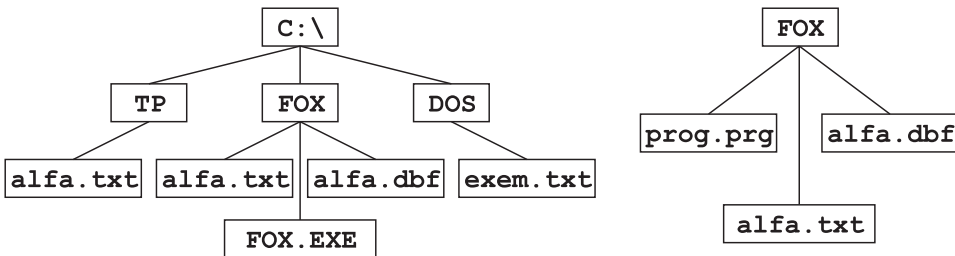
```
! md exemple
set default to exemple
! copy con exempl.txt
copy file exemp.txt to exe.txt
copy to texte
delete file exempl.000
modify command \program

dir \fox26\programe\*.prg
!copy *.txt \texte
!copy file elevi.dbf to \*.dbf
!mkdir \bazedate
cd \bazedate
!rd exemple
? curdir()
```

II. Prin ce secvență se poate realiza mutarea documentului `tabel.txt` din directorul `\UTIL` al discului C în rădăcina discului A sub numele `lista.doc`?

- `rename c:\util\tabel.txt to a:`
- `copy file c:\util\tabel.txt to a:\lista.doc`
- `delete file c:\util\tabel.txt`
- `move c:\util\tabel.txt a:\lista.doc`

III. Fie structura discurilor C și A din figura următoare.



- Știind că suntem în directorul `C:\`, lansați programul `C:\FOX\FOX.EXE` fixând `C:\TP` ca director curent.
- Creați un subdirector nou sub `ALFA` cu numele `GAMA`, care să conțină toate fișierele din `ALFA`.
- Afișați conținutul directorului curent.
- Afișați conținutul fișierului `ALFA.TXT`.
- Afișați toate fișierele cu extensia `.prg` din directorul `C:\FOX`.
- Fixați ca director de lucru `C:\DOS` și apoi afișați numai bazele de date.
- Copiați fișierul `ALFA.TXT` în `C:\TP`.
- Lansați programul `A:\PROG.PRG`.
- Ștergeți toate fișierele cu extensia `.TXT` din directorul `TP`.
- Mutați fișierul `A:\PROG.PRG` în directorul rădăcină al discului C.

Tipuri de date în Visual FoxPro și operații specifice

- Tipul numeric
- Tipul șir de caractere
- Tipul dată calendaristică
- Tipul logic
- Comenzi legate de manevrarea variabilelor și a expresiilor

Principalele tipuri de date cu care lucrează FoxPro sunt datele numerice, șirurile de caractere, datele logice și datele calendaristice. Datele de tip memo sau de tip general pot fi definite numai în contextul bazei de date, iar operațiile cu aceste tipuri vor fi prezentate în lecțiile următoare.

Tipul numeric

Este folosit în diferite variante, alese în funcție de domeniul de valori în care se va încadra variabila sau câmpul de acest tip.

Varianta	Descriere
Numeric	Întregi sau fracții. Ocupă 8 B în memorie. Până la 20 B în tabela de date. Este folosit pentru valori între -0.9E19 și +0.9E19.
Float	în VMSP pe 4 B
Double	Pentru valori între -4.9E-324 și 1.79E308 în VMSP pe 8 B.
Integer	Pentru valori întregi între -2147483647 și +2147483646 pe 4 B.
Currency	Valori monetare între -9*10**14 și +9*10**14; pe 8 B.

Operatori

Aritmetici	+, -, *, /, ** (ridicare la putere), % (modulo)
Relaționali	<, >, <=, >=, # sau <> sau != (diferit), = (egal)

Funcții standard matematice uzuale

Funcție	Efect
=ABS (n)	Valoarea absolută. <i>Exemplu:</i> ABS (-15) =15.
=ROUND (n1 , n2)	<n1> este rotunjit la zecimala al cărei ordin e dat de <n2> <i>Exemplu:</i> ROUND (1 . 7567 , 3) =1 . 757;
=MOD (n1 , n2)	Restul împărțirii întregi a lui n1 la n2. <i>Exemplu:</i> MOD (5 , 2) =1
=INT (n)	Partea întreagă a lui <n>. <i>Exemplu:</i> INT (7 . 25) =7; INT (7 . 87) =7

=CEILING (n)	Aproximare la cel mai mic întreg mai mare sau egal cu <n>. <i>Exemple:</i> CEILING (5.87)=6; CEILING (-5.87)=-5
=FLOOR (n)	Aproximare la cel mai mare întreg mai mic sau egal cu <n>. <i>Exemplu:</i> FLOOR (5.87)=5; FLOOR (-5.87)=-6
=STR (n1 [, n2 [, n3]])	Conversie la șir a lui n1; n2 este lungimea; n3 este numărul de poziții zecimale. <i>Exemplu:</i> STR (1432.456, 12, 4)="1432.456"

Tipul șir de caractere

Acest tip este folosit pentru variabilele sau câmpurile care conțin caractere ASCII delimitate prin apostrof, ghilimele sau paranteze pătrate.

Exemplu: "Scarlet O'Hara"

Operatori

De concatenare + (concatenarea șirurilor);

Exemplu: **[buna]+'dimineata'='buna dimineata'**

- (concatenare cu mutarea spațiilor de la sfârșitul primului șir la sfârșitul șirului rezultat

Exemplu: **[buna]-'dimineata'=[bunadimineata]**

Relaționali <, <=, >, >=, =, # sau !=, sau <> (diferit) \$ (inclus)

Exemple: **[alb] = 'albastru'** returnează adevărat pentru

SET EXACT OFF; [alb] = 'albastru' returnează fals

pentru **SET EXACT ON; [alb] \$ 'albastru'=adevarat**

Funcții uzuale aplicate șirurilor

Funcție	Efect
=SUBSTR (s , n1 , n2)	Extrage un subsșir din șirul <s> începând cu caracterul de pe poziția <n1>, de lungime <n2>. <i>Exemplu:</i> SUBSTR ('ABCD' , 2, 2)='BC'
=LEFT/RIGHT (s , n)	Extrage primele/ultimele <n> caractere din șirul <s>. <i>Exemple:</i> LEFT ("ABCD" , 2)="AB" RIGHT ('ABCD' , 2)='CD'
=LEN (s)	Returnează lungimea șirului <s>. <i>Exemplu:</i> LEN ('ALFA')=4
=LTRIM/RTRIM/ ALLTRIM (s)	Elimină spațiile de la stânga șirului (LTRIM), de la dreapta șirului (RTRIM) sau din ambele părți (ALLTRIM). <i>Exemplu:</i> LTRIM (' MIA ')="MIA "
=AT (s1 , s2)	Returnează poziția șirului <s1> în <s2>. <i>Exemplu:</i> AT ('NR' , 'str Ploii nr 5')=0;
=LOWER/UPPER/ PROPER (s)	Transformă șirul în minuscule/majuscule/tip titlu. <i>Exemple:</i> LOWER ('VARA')='vara' UPPER ('galben')='GALBEN' PROPER ('ana maria')='Ana Maria'

=VAL (s)	Realizează conversia unui șir la număr. <i>Exemplu:</i> VAL ('1433.44')=1433.44
=OCCURS (s1 , s2)	Numără aparițiile lui <s1> în șirul <s2>. <i>Exemplu:</i> OCCURS ("A" , "ALFA")=2

Tipul dată calendaristică

Tipul Date este folosit pentru gestionarea datelor calendaristice reprezentate pe 8 B (an, lună, zi). Forma de reprezentare diferă în funcție de formatul implicit sau setat de utilizator.

Exemplu: {01/31/94} se reprezintă 31 ianuarie 1994 în format american.

Tipul DateTime este folosit pentru a memora valori ce conțin atât data, cât și ora. Regulile sunt aceleași ca la tipul Date.

Exemplu: {12/12/1999 10:03:09}

Comenzi utile

SET CENTURY ON/OFF	Specifică anul, inclusiv secolul.
SET DATE GERMAN/AMERICAN/. . .	Specifică formatul de dată.

Operatori

Aritmetici	+ (adună) sau - (scade) un număr de zile la o dată <i>Exemplu:</i> {01.01.2001}+3={04.01.2001}
Relaționali	<, <=, >, >=, =, <>, #, != <i>Exemplu:</i> {01.01.2001} < {12.31.2001}=T.

Funcții standard pentru date calendaristice

Funcție	Efect
=DATE () /TIME () / DATETIME ()	Returnează data/ora curentă a sistemului. <i>Exemple:</i> DATE ()={01/01/94} ; time ()="10:12:00"; datetime ()={08/05/99 07:18:20PM}
=DAY/MONTH/ YEAR (d)	Extrage numărul zilei/lunii/anului din data <d>. <i>Exemplu:</i> Fie x={01.07.1997} cu formatul zz.ll.aaaa. Atunci DAY (x)=1; MONTH (x)=7; Year (x)=1997.
=CMONTH (d)	Returnează numele lunii din data <d>. <i>Exemplu:</i> CMONTH ({01/07/94})='JANUARY'
=DTOS/DTOC (d)	Returnează data sub formă de șir. <i>Exemplu:</i> Fie x={01.07.94}. DTOS (x)="19940701"; DTOC (x)="01/07/94"
=CTOD (s)	Realizează conversia unui șir la dată calendaristică. <i>Exemplu:</i> CTOD ("01.01.94")={01.01.94}.

Tipul logic

Acest tip este folosit pentru variabile sau câmpuri ce pot avea doar două valori, notate .T. (true=adevărat) și .F. (false=fals).

Operatorii logici sunt **OR**, **AND**, **NOT** sau !

Alte funcții uzuale aplicate tuturilor tipurilor de date

Funcție	Efect
<code>=MAX/MIN (e1 , e2 [, e3 . . .])</code>	Calculează extremul dintre valorile <e1>, <e2>,... <i>Exemple:</i> <code>max (14 , 3)=14</code> ; <code>min (14 , 3 , 15 , 6)=3</code>
<code>=TYPE (eC)</code>	Returnează litera corespunzătoare tipului de dată. <i>Exemplu:</i> <code>TYPE ("12")=N</code> ; <code>TYPE (" [12] ")=C</code>
<code>=IIF (eL , e1 , e2)</code>	Dacă <code>e1=.T.</code> returnează <code>e1</code> , altfel returnează <code>e2</code> . <i>Exemplu:</i> <code>IIF (3=5 , "corect" , "incorect")="incorect"</code> .
<code>=BETWEEN (e1 , e2 , e3)</code>	Testează dacă <code>e1</code> aparține intervalului (<code>e2,e3</code>). <i>Exemplu:</i> <code>BETWEEN (3 , 0 , 20)=.T.</code>
<code>=EMPTY (e)</code>	Testează dacă expresia dată ca parametru este vidă. <i>Exemple:</i> <code>Empty ("")=.t.</code> ; <code>empty ({})=.t.</code> ; <code>empty (5)=.f.</code>
<code>INLIST (e1 , e2 [, e3 . . .])</code>	Testează dacă <code>e1</code> aparține listei date de următorii parametri. <i>Exemple:</i> <code>inlist (2 , 1 , 2 , 3 , 4)=.t.</code> ; <code>inlist ('d' , 'a' , 'b' , 'c')=.f.</code>

Gestiunea variabilelor

O variabilă de memorie reprezintă o modalitate prin care FoxPro depozitează și utilizează temporar date într-un program. **Variabilele utilizator** sunt zone de memorie cărora li se atribuie un nume, un tip și o valoare. Numele variabilei este un șir de caractere alfanumerice. Tipul variabilei este atribuit acesteia odată cu valoarea.

Tablourile sau masivele de date sunt structuri statice *neomogene*. Pot avea cel mult două dimensiuni.

Declararea tablourilor

```
DIMENSION / DECLARE <tablou (dim1 [ , dim2] )> , .>
```

Inițializarea unui tablou se face odată cu declararea dimensiunii la valoarea .F. Altă inițializare globală a tuturor elementelor unui tablou se poate face prin comanda **STORE**. O matrice se poate redimensiona fără ca prin această schimbare să i se reinițializeze componentele.

Exemplu:

```
dimension a[2 , 2]
store 5 to a      && o matrice pătrată 2x2 este inițializată pe valoarea 5
dimension a[3 , 3]  && redimensionare
?a[1,1] , a[1,2] , a[1,3] , a[2,1] , a[2,2] , a[2,3] , a[3,1] , a[3,2] , a[3,3]
```

```
5 5 f. 5 5 .f. .f. .f. .f.
```

Funcții standard aplicate tablourilor

Funcție	Efect
<code>=alen(t, n)</code>	Definirea dimensiunii tabloului ca număr total de elemente (dacă $n=0$), număr de linii (dacă $n=1$), sau număr de coloane (dacă $n=2$).
<code>=ains(T, P[, N])</code>	Inserare în tabloul T pe poziția P a unui element (dacă lipsește N), a unei linii (dacă $N=1$), a unei coloane (dacă $N=2$).
<code>=adel(T, P[, N])</code>	Ștergerea unui element (dacă lipsește N), a unei linii (dacă $N=1$), a unei coloane (dacă $N=2$).
<code>=ascan(T, E[, P[, L]])</code>	Returnează poziția unei expresii E prin căutarea în tabloul T, începând de la poziția P pe lungimea L.
<code>=acopy(T1, T2[, P[, L]])</code>	Copierea elementelor din T1, începând cu cel de pe poziția P pe lungime L în tabloul T2.
<code>=asort(T, [P[, L[, N]])</code>	Sortarea unui tablou T în mod crescător (dacă $N=0$) sau descrescător (dacă $N\neq 0$), începând cu elementul de pe poziția P pe lungimea L.

Exemplu

Fie tabloul B, cu valorile (1, 5, 7, 9) și tabloul A cu valorile ((11, 12, 13, 14), (21, 22, 23, 24), (31, 32, 33, 34)).

Sarcina	Rezolvare
Inserarea unei linii pe poziția 3 din A	<code>=ains(a, 3)</code>
Inserarea unui element pe poziția 4 din B	<code>=ains(b, 4)</code>
Inserarea unei coloane pe poziția 1 din A	<code>=ains(a, 1, 2)</code>
Ștergerea elementului al 4-lea din B	<code>=adel(b, 4)</code>
Ștergerea liniei 3 din A	<code>=adel(a, 3)</code>
Ștergerea coloanei 2 din A	<code>=adel(a, 2, 2)</code>

Folosirea funcției de sortare a tablourilor

Fie B(3,5,7,1,10) și A((11,12,13,14),(21,22,23,24),(31,32,33,34)).

Comandă	Efect
<code>=asort(b)</code>	B(1, 3, 5, 7, 10) crescător întregul tablou
<code>=asort(b, 1, 5, 5)</code>	B(10, 7, 5, 3, 1) descrescător întregul tablou
<code>=asort(b, 1, 3, 1)</code>	B(7, 5, 3, 1, 10) descrescător, primele 3 elemente
<code>=asort(b, 3, alen(b), 0)</code>	B(3, 5, 1, 7, 10) crescător, începând de la poziția 3
<code>=asort(a, 3, alen(z), 1)</code>	sortare descrescătoare după coloana 3
<code>=asort(a, 3)</code>	sortare crescătoare după coloana 3
<code>=asort(a, 9)</code>	sortare crescătoare a liniilor 3 și 4 după valorile coloanei 2

Variabilele pot fi **locale (private)** unei unități funcționale sau **globale (publice)**, recunoscute de toate unitățile funcționale subordonate.

Declararea variabilelor publice:

```
PUBLIC [ARRAY] <lista-var>
```

Dacă variabilele sunt tablouri, se specifică clauza **ARRAY**.

Declararea variabilelor private:

```
PRIVATE/LOCAL <lista-var>/ALL LIKE/EXCEPT <sablon>
```

Clauza **ALL LIKE** permite declararea privată a tuturor variabilelor care verifică un anumit **<sablon>**. Clauza **ALL EXCEPT** declară ca locale toate variabilele definite în modul, cu excepția celor care verifică **<sablon>**.

Comenzi de lucru cu variabilele

Comanda	Efect
<code><var>=<expresie></code> <code>/STORE <exp> TO <lista-var></code>	Se atribuie variabilei valoarea expresiei. Comanda Store permite atribuirii multiple.
<code>INPUT [<mesaj>] TO <var></code>	Comadă de citire. Variabila primește valoarea și tipul expresiei introduse de la tastatură.
<code>ACCEPT [<mesaj>] TO <var></code>	Comanda de citire a unei variabile de tip caracter.
<code>@ <r,c> GET<var></code> <code>[PICTURE <șablon>]</code> <code>[DEFAULT<exp>] [VALID <cond>]</code> <code>READ</code>	Comanda de editare a unei variabile sau câmp al unei tabele la execuția comenzii READ . <code><r, c></code> specifică poziția pe ecran a variabilei. PICTURE specifică un șablon de editare; DEFAULT este valoarea inițială; VALID este condiția de validare.
<code>? <lista-expresii> AT <col></code>	Permite afișarea expresiilor pe linia curentă, începând din coloana specificată.
<code>@ <r,c> SAY <exp></code>	Afișează valoarea expresiei <code><exp></code> începând din punctul <code><r, c></code> .
<code>RELEASE <lista_var> /</code> <code>ALL [LIKE/EXCEPT <șablon>]</code>	Ștergerea unei liste de variabile sau a tuturor sau a celor ce verifică un șablon.
<code>SAVE TO <fis.mem></code> <code>[ALL LIKE/EXCEPT <șablon>]</code>	Salvarea pe disc în fișierul <code>.mem</code> a variabilelor.
<code>RESTORE FROM <fis.mem></code> <code>[ADDITIVE]</code>	Restaurarea prin suprascriere sau adăugare a variabilelor din fișierul <code>.mem</code> în zona de lucru.
<code>WAIT [<mesaj>] [TO <var>]</code> <code>[WINDOWS] [NOWAIT]</code>	Pauză în program cu afișarea unui mesaj (eventual într-o fereastră); tasta apăsată este transmisă variabilei <code><var></code> .
<code>=MESSAGEBOX (<mesaj>)</code>	Funcție pentru afișarea unui mesaj.

Lucrul cu imprimanta

Sunt disponibile două comenzi care direcționează efectul comenzilor de afișare către imprimantă.

Comanda	Efect
<code>SET PRINTER ON/OFF</code>	Activarea/dezactivarea imprimantei.
<code>SET DEVICE TO PRINTER/ TO SCREEN</code>	Direcționarea comenzii către imprimantă sau ecran.

Exemple

Comenzi	Efect
<code>store 0 to a, b, c b='alfa'</code>	Variabilele numerice a, b, c sunt inițializate cu valoarea zero; este modificat conținutul și tipul variabilei b.
<code>? "a=",a</code>	Se afișează a=0.
<code>x={01. 09. 95}</code>	Se creează variabila x de tip dată calendaristică.
<code>@ 1,5 say "data="+dtoc(x)</code>	Se afișează din punctul 1,5 expresia data=01091995 .
<code>input "nume?" to nume</code>	Pentru citirea unui șir trebuie dați și delimitatorii acestuia.
<code>accept "nume?" to nume accept "varsta?" to v</code>	Nu mai este nevoie de delimitatori. Atenție! Vârsta va fi tot de tip caracter.
<code>release all like a*</code>	Se șterg variabilele care încep cu a.
<code>save to beta all like a</code>	Se salvează în fișierul beta.mem variabila a.
<code>@ 2,2 get b picture '99.99' default 2.00 valid b<=10.00 read</code>	Se afișează o zonă de editare și se așteaptă introducerea valorii pentru variabila b; inițial variabila b=2.00; nu se părăsește zona de editare dacă valoarea introdusă este mai mare ca 10.
<code>set printer on ? 'text'</code>	Se direcționează efectul comenzii de afișare ? către imprimantă.
<code>Set device to printer Set printer on @ 1,5 say 'text'</code>	Se direcționează efectul comenzii de afișare @ către imprimantă.

Macrosubstituție și expresii nume

Utilizarea conținutului unei variabile într-o comandă ca identificator se poate face prin mecanismul numit **macrosubstituție**. Indicarea acestei operații se face prin simbolul „&” (ampersand) pus înaintea numelui variabilei. În locul macrosubstituției se poate folosi **expresia-nume** (denumită astfel pentru că poate fi folosită oriunde în FoxPro; așteaptă un nume, un identificator). Expresiile-nume pot conține nume de fișiere, nume de ferestre, nume de aliasuri, numere de zone etc.

Exemple

<code>Alfa='elevi'</code>	Se atribuie variabilei alfa șirul ' Elevi '.
<code>Beta='alfa'</code>	Se atribuie variabilei Beta șirul ' alfa '.
<code>? beta</code>	Se va afișa conținutul lui beta , adică ' alfa '.
<code><u>alfa</u></code>	Se va înlocui variabila beta cu valoarea sa, alfa și se va afișa conținutul acesteia, adică ' elevi '.
<code>? &beta</code>	
<code><u>elevi</u></code>	
<code>use alfa</code>	Se va deschide fișierul ' alfa '.
<code>use &alfa</code>	Se va deschide fișierul ' elevi '.



sarcini de laborator

1. Urmăriți efectul funcției **STR** și răspundeți la întrebări:

```
a=1234
b=mod(a,23)
lung=5
zecim=3
? str(a, lung, zecim)
? str(a, 3)
? str(b, lung)
? str(b, 10, zecim)
? str(b, lung)
```

- Ce se întâmplă dacă numărul are zecimale și parametrul al treilea lipsește?
- Ce se întâmplă dacă lungimea indicată este prea mare? Vor fi adăugate spații?
- Cum trebuie calculată lungimea pentru a reprezenta și zecimalele?
- Ce se întâmplă dacă expresia are zecimale, dar lipsește în funcție parametrul al treilea? Are loc truncierea? Se rotunjește la întreg?

2. Urmăriți rezultatul afișat în fiecare caz. Ce efect are comanda **SET EXACT**?

```
a='abcd--'      ?a<c, a=c, a>c      SET EXACT OFF
b='ef--gh'      ?b<d, b=d, b>d      ?b<e, b=e, b>e
c=a+b+'*'      ?a>f, a=f, a<f      ?a=f, f=a, a#f, f#a
d=a-b+'*'      ?a>d, a=d, a<d      ?'abcd' $f
?c, d          SET EXACT ON       ?f$a, b$a, a$b
e=b-a+'*'      ?a<c, a=c, a>c      ?g$a, a$"ABCD
f='bc'         ?b<d, b=d, b>d
```

3. Urmăriți efectul următoarelor comenzi și scrieți ce se afișează:

```
? min({ 06/09/93}, {05/09/93})      set century on      ? cmonth(x)
? x=date()                            ? year(x), x        ? cdow(x)
? day(x), month(x), year(x)          ? dmy(x)            ? len(month(x))
? mdy(x)                              ? ymd(x)
```

4. Ce realizează secvența următoare?

```
x=time()
h=val(left(x, 2))
m=val(substr(x, 4, 2))
s=val(right(x, 2))
y=str(h,2) + ":" + iif(len(alltrim(str(m)))=1, 0'+;
str(m, 1), str(m, 2))+ ":" + iif(len(alltrim(str(s)))=1, "0"+;
str(s, 1), str(s, 2))
```

5. Pentru o dată calendaristică primită din exterior sub forma luna/zi/an prin instrucțiunea **ACCEPT** să se afișeze sub formatul **anul:ziua:luna**.

6. Pentru o dată calendaristică primită din exterior sub forma luna/zi/an prin instrucțiunea **INPUT** să se afișeze sub formatul **anul:ziua:luna**.

7. Introduceți în variabila **AZI** data curentă și afișați numele zilei de azi, de ieri și de mâine.

8. Fie **MATRICOL="10A25"**. Extrageți în variabile distincte informațiile anul, litera și numărul de ordine. Afișați o expresie, recompunând matricolul.

9. Care sunt funcțiile necesare obținerii următoarelor valori numerice plecând de la șirurile $x="12.5"$ și $y=".145"$?

⇒ 12.145 ⇒ 14.512 ⇒ -12 ⇒ 0.145
 ⇒ 145.12 ⇒ 0.12 ⇒ 157 ⇒ 12.000

10. Care sunt funcțiile care permit afișarea numărului $N=145.567$ sub forma următoarelor șiruri:

⇒ "145.567" ⇒ " 145.57" ⇒ "145.56" ⇒ "0.14"
 ⇒ " 146" ⇒ "145" ⇒ "291" ⇒ "14.567"
 ⇒ " 145.6" ⇒ "-14" ⇒ "567" ⇒ "0.00145"

11. Știind că la o stație PECO s-a format o coadă de n mașini și că timpul de servire cu benzină a unei mașini este în medie X minute, aflați cât timp va aștepta ultima mașină în zile, ore și minute.

12. La un magazin trebuie să se achite suma S în hârtii de valoare 10000, 5000 sau 1000 unități monetare (u.m.). Care va fi numărul minim de bancnote de fiecare valoare pentru acoperirea sumei și ce rest (în monede de 100 u.m.) primește clientul?

Exemplu: Dacă $S=126400$, clientul va plăti $12*10000+1*5000+2*1000$ și va primi rest 600 lei.

13. Se dau trei numere, reprezentând laturile unui triunghi. Afișați ce fel de triunghi este: dreptunghic, ascuțitunghic, obtuzunghic.

Urmăriți rezolvările propuse și corecții – dacă este cazul – erorile!

**varianta I* && a, b, c sunt date numerice citite deja!*

```
x=a^2
y=b^2
z=c^2
?iif(x=y+z or y=x+z or z=x+y, 'drept', iif(x<y+z and y<x+z and z<x+y, 'ascutit', iif(x>y+z or y>x+z or z>x+y, 'obtuz', '')))
```

varianta a II-a

```
lmax=max(a, b, c)
lmin=min(a, b, c)
lmijl=iif((lmax=a and lmin=b) or (lmax=b and lmin=a), ;
c, iif((lmax=a and lmin=c) or;
(lmax=c and lmin=a), b, a))
l1=lmax^2
l2=lmin^2
l3=lmijl^2
?iif(l1=l2+l3, 'drept', iif(l1<l2+l3, 'ascutit', 'obtuz'))
```

Încercați să vedeți efectele noilor funcții standard care apar prin asistentul Help!

varianta a III-a

```
a1=acos((b^2+c^2-a^2)/(2*b*c))
a2=acos((a^2+c^2-b^2)/(2*a*c))
a3=acos((a^2+b^2-c^2)/(2*a*b))
x=max(a1, a2, a3)
?iif(x=pi()/2, 'drept', iif(x<pi()/2, 'ascutit', 'obtuz'))
```

varianta a IV-a

```
lmax=max(a, b, c)
l2l3=iif(max=a, 'bc', iif(max=b, 'ac', 'ab'))
l2s=left(l2l3, 1)
l3s=right(l2l3, 1)
l2=evaluate(l2s)
l3=evaluate(l3s)
?iif(lmax^2=l2^2+l3^2, 'drept', iif(lmax^2<l2^2+l3^2, 'ascutit', 'obtuz'))
```

Crearea tabelelor

- Definirea structurii unui tabel ¹
- Modificarea structurii conceptuale a unui tabel
- Editarea câmpurilor a unui tabel
- Popularea (încărcarea) tabelului cu date
- Deschiderea și închiderea unui tabel
- Filtrarea structurii sau selectarea câmpurilor
- Filtrarea articolelor

Definirea structurii unui tabel

Produsul FoxPro are la bază teoria algebrelor relaționale, dar identifică noțiunea de relație cu cea de fișier, atributele cu domeniile și cu câmpurile, iar tuplele cu articolele.

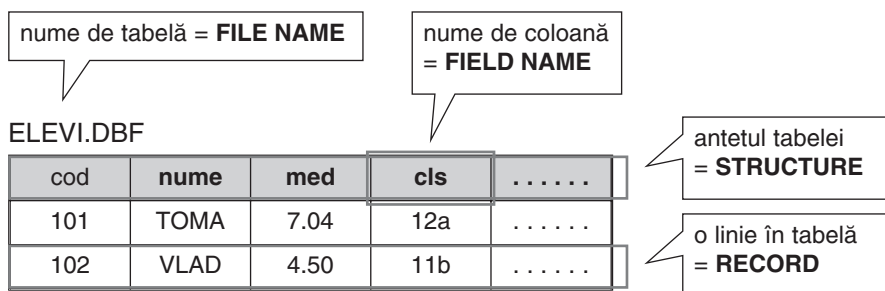


Figura 5-1: Structura unui tabel

În acest fel, **tabela (relația)** este definită ca fișier secvențial cu extensia .DBF, având articole de lungime fixă și, ca primă înregistrare, un antet care reține structura tabelii. Celelalte articole conțin datele propriu-zise ale tabelii.

O bază de date este definită ca o colecție de tabele relaționate, împreună cu indecșii, procedurile stocate și imaginile asociate tabelilor. Extensia acestui fișier este .DBC.

Proiectarea structurii unei tabele se poate realiza direct, prin definirea directă a câmpurilor în cadrul comenzii **CREATE TABLE**, sau interactiv, prin comanda-ecran **CREATE**.

a. Comanda CREATE permite proiectarea interactivă a structurii. Din meniul sistem, selectați **File, New, Table**. Apare fereastra de dialog Table Designer. Fereastra are trei secțiuni (tab-uri), dintre care vom lucra deocamdată doar cu tab-ul **Fields**, care permite definirea informațiilor de structură. Astfel:

- a) Name – permite introducerea numelui câmpului – până la 10 caractere;
- b) Type – permite introducerea tipului de dată al câmpului – poate fi numeric, caracter, logic, dată calendaristică, Memo, General etc.;
- c) Width – permite specificarea altei lungimi a câmpului decât cea implicită;
- d) Decimal – permite introducerea numărului de poziții zecimale;

¹ Sunt folosite și acceptate în domeniu denumirile tabel și tabelă, convenție pe care o urmăm și în manual.

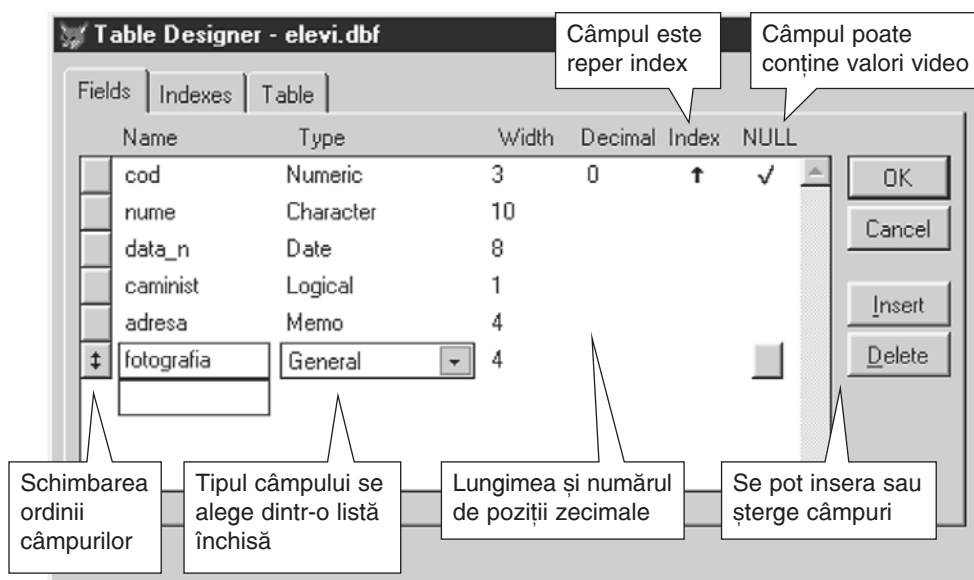


Figura 5-2: Fereastra Table Designer

- e) Index – permite crearea unui index având câmpul curent drept cheie;
- f) NULL – este un comutator pentru acceptarea sau nu a valorilor nule.

b. Comanda CREATE TABLE permite specificarea directă a structurii:

```
CREATE TABLE/ DBF <fis.dbf> (<lista-definitii>)
```

```
<definitie>:=<nume-câmp> <tip>[[<lungime>], <zecimale>]]
```

Exemplu: Fie tabela **ELEVI.DBF**

nume	pren	cls	camin	absn	med	adr	dn	sit
c, 20	c, 20	c, 3	l, 1	n, 2	n, 5, 2	m, 10	d, 8	c, 10

Comanda de creare a acestei tabele este:

```
create dbf elevi (nume C(20), pren C(20), cls C(3), camin L,
absn N(2), med N(5,2), adr M, dn D, sit C(10))
```

Exemple

Proiectați structura unei tabele care să permită răspunsuri la diferite întrebări de felul celor din prima coloană. Observați în coloana a doua modalitatea de analiză pentru stabilirea structurii.

Interogări

Analiza

1. Care sunt elevii școlii, în ordine alfabetică?

Tabela va trebui să conțină numele și prenumele fiecărui elev – ca un singur câmp, de tip **Caracter**. Pentru obținerea informațiilor ordonate alfabetic, trebuie precizat câmpul drept criteriu de indexare!

2. Cine locuiește la cămin? Care este adresa stabilă a căminiștilor?

Putem specifica printr-un câmp **Logic** dacă elevul este sau nu căminist. Pentru adresă vom folosi tipul **Memo**.

3. Care sunt elevii care își sărbătoresc azi ziua de naștere și în ce clasă sunt ei?	Pentru rezolvare va trebui să reținem data nașterii (tip dată-calendaristică) a fiecărui elev și clasa (tip caracter).
4. Care este situația școlară a elevului „POPA ION” (promovat, corigent, repetent)?	Putem reține mediile la toate obiectele, iar comparând aceste medii cu 5 să obținem situația cerută. Dar observăm că nu există altă referire la mediile pe obiecte (sunt și multe!), deci vom putea reține informația solicitată ca text.
5. Care este cea mai mare medie?	Prelucrarea tabelului va determina maximul dintre valorile unui câmp. Media de absolvire va fi definită ca tip numeric cu 2 zecimale .
6. Cât la sută din totalul absențelor sunt nemotivate?	Interogarea necesită reținerea pentru fiecare elev a absențelor nemotivate și motivate (tip întreg) și exprimarea procentuală a raportului dintre totalul absențelor și numărul celor nemotivate.
7. Sunt mai mulți elevi cu numele „Popa Ion”?	Tabela va conține pentru fiecare elev drept <i>cheie unică</i> numărul matricol (tip întreg).
8. Cum arată șeful clasei a X-a A?	Pentru a reține fotografiile elevilor, vom proiecta un câmp de tip General . Prelucrarea va determina elevul cu media cea mai mare din clasa a X-a și va afișa fotografia sa.

Observație. Proiectarea structurii trebuie să se facă analizând cererile de informații în totalitatea lor și fixând tipul fiecărei date care va fi reținută. Se va deschide apoi ecranul de proiectare și se vor introduce informațiile de structură. Activitatea de analiză este extrem de importantă. Unele cereri de informații sunt date explicit de client, altele trebuie să fie imaginate de către proiectant.

Modificarea structurii conceptuale a unui tabel

O tabelă poate fi modificată prin aceeași fereastră Table Designer, deschisă la comanda **MODIFY STRUCTURE** sau selectând **View, Table Designer**. Utilizatorul poate șterge, adăuga sau insera câmpuri, poate modifica lungimea sau tipul unor câmpuri. Datele existente vor fi copiate în noua structură prin verificarea numelui câmpului din cele două structuri. În cazul în care coincid, datele vor trece în noua structură, făcându-se conversia la noul tip de câmp, dacă este posibil.

Atenție! Modificarea structurii poate duce la pierderea datelor.

Editarea câmpurilor unui tabel

Folosiți comenzile **CHANGE/EDIT** pentru afișarea câmpurilor tabelului în scopul editării.

Popularea (încărcarea) tabelului cu date

Utilizatorul poate introduce date în fișierul **.DBF** imediat după salvarea structurii, răspunzând afirmativ la întrebarea sistemului: Input data records now? (Introduceți acum date?). O altă posibilitate de populare a tabelului este dată de comanda **APPEND**. Datele sunt introduse conform tipului și lungimii declarate la proiectare.

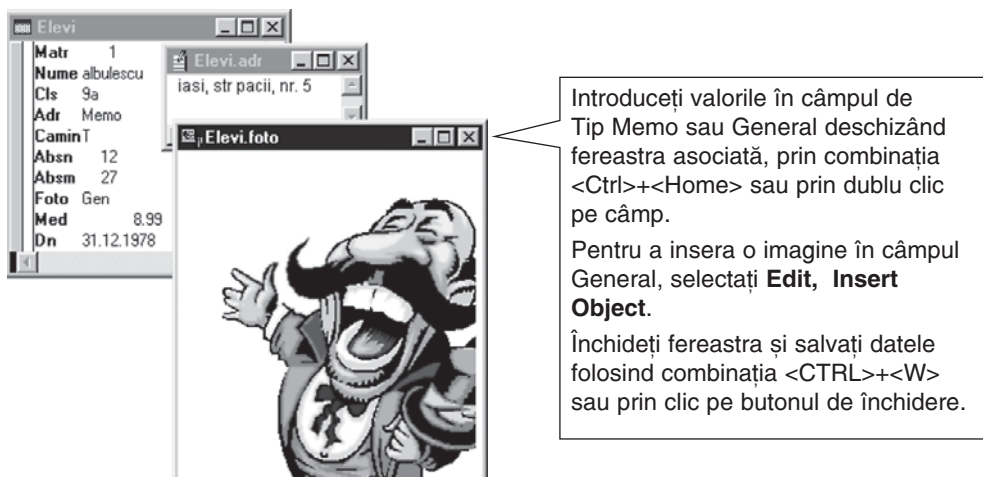


Figura 5-3: Popularea tabelului

Deschiderea și închiderea unui tabel

Orice operație asupra unei tabeli, cu excepția definirii structurii, impune deschiderea acesteia, iar după terminarea activității asupra datelor respective, tabela trebuie închisă. Deschiderea unui fișier tabelă se face de către sistemul FoxPro într-o zonă de memorie numită zonă de lucru (*work area*). Într-o zonă de lucru se poate deschide o singură tabelă.

Deschiderea unei tabeli se poate face prin comanda **USE** în care se precizează numărul zonei în care se deschide fișierul (clauza **IN <zona>**).

```
USE <fis.dbf> IN <zona> [ALIAS <nume-alias>] [AGAIN]
```

Tot prin comanda **USE** se poate asocia și un alias (un pseudonim, o prescurtare) în vederea unei referiri mai clare atât a câmpurilor, cât și a zonei în care s-a deschis fișierul (clauza **ALIAS**). Clauza **AGAIN** permite deschiderea aceluiași fișier în două zone.

Închiderea unei singure tabeli se face prin comanda **USE IN<zona>**.

Închiderea tuturor tabelilor, indiferent de zona unde au fost deschise, se face prin comanda **CLOSE ALL**. Zona de lucru care va deveni activă se precizează prin comanda: **SELECT <zona>/<nume-alias>**.

Zona este precizată printr-un număr (sau o literă de la A la I) sau prin aliasul fișierului deschis în zonă. Ultima zonă selectată se numește *zonă curentă* și toate referirile implicite se vor face la aceasta. Comanda **SELECT 0** deschide prima zonă liberă.

Calificarea câmpurilor și a variabilelor

Atunci când sunt deschise mai multe fișiere sau când există variabile de memorie cu același identificator ca al câmpurilor, se folosește operația de calificare:

```
<alias>.<nume-câmp> și <M>.<nume-variabila>.
```

Exemple

<code>Select 2</code>	Zona 2 va conține tabela Profesori.dbf.
<code>use profesori</code>	
<code>modify structure</code>	Modificarea structurii fișierului din zona curentă.
<code>use elevi in 1 alias EL</code>	Se deschide Elevi.dbf în zona 3 cu aliasul EL.
<code>Select EL</code>	Poziționare pe zona 1 prin aliasul EL.
<code>select 0</code>	Deschiderea primei zone libere.
<code>create student</code>	Crearea fișierului Student.dbf în zona liberă.
<code>sele student</code>	Selectarea zonei prin aliasul fișierului.
<code>accept "nume ?" to Nume</code>	Variabila Nume este creată prin citire.
<code>? EL.Nume, m.Nume</code>	Afișarea valorii câmpului Nume din primul articol și a valorii citite în variabila cu același identificator.

Filtrarea structurii sau selectarea câmpurilor

Uneori este necesar ca o parte din câmpuri să fie active. Comenzile care operează asupra tabelor folosesc de obicei **clauza Fields** în care este precizată lista câmpurilor active. O altă posibilitate este crearea unei liste de câmpuri (din una sau mai multe tabele) de care se va ține seama sau nu.

```
SET FIELDS TO <lista-câmpuri> [/R] / ALL LIKE / EXCEPT <sablon>
```

Ștergerea listei de câmpuri și anularea comenzii de selecție se fac prin:

```
SET FIELDS TO . Activarea listei de câmpuri reținută anterior sau dezactivarea ei se fac prin: SET FIELDS ON/OFF .
```

Filtrarea articolelor

Operația de filtrare a articolelor care vor suporta efectul unei comenzi este realizată prin clauzele: **<domeniu>**, **FOR**, **WHILE**. Dacă aceeași selecție va fi folosită de mai multe ori, este utilă comanda:

```
SET FILTER TO [<cond>]
```

Comanda permite accesul numai la articolele care îndeplinesc condiția **<cond>**. Spunem că se **deschide un filtru** pe baza de date, care va fi activ până când se închide baza de date (prin **USE**, **CLOSE DATABASES**) sau se închide filtrul prin **SET FILTER TO**.

BIBLIOTECA

Reluăm activitatea de proiectare a structurii tabelelor, ținând cont de informațiile solicitate și rezolvarea cererilor.

1. Care sunt cărțile existente (titlu, autor, editura, editia, preț)?	Să presupunem că biblioteca are un singur exemplar dintr-o carte, deci putem lua drept cheie numărul de inventar. Tabela Carti va cuprinde titlul, numele autorului, editura (câmpuri tip șir de caractere), iar ediția și prețul vor fi date numerice.
2. Ce cărți au intrat sau au ieșit după 1994?	Completăm structura Cărti cu atributele an-intrare și an-iesire de tip Dată calendaristică.
3. Lista cărților din colecția „De dragoste“.	Este necesar în tabela Carti atributul colectie de tip șir de caractere.
3. Lista cititorilor înscriși la bibliotecă mai mici ca 15 ani.	Este necesar în tabela Cititori atributul data-nașterii, de tip dată calendaristică.
4. Sunt și profesori de la Liceul de Informatică înscriși la bibliotecă?	Atributele ocupație și loc de muncă vor fi incluse în tabela Cititori. Le vom defini ca șiruri de caractere.
5. Sunt cititori restanțieri?	Vom filtra tabela Operații pentru data restituirii necompletată și numărul de zile de la data împrumutului peste 14. Dacă numărul de articole accesibile este diferit de zero afișăm „da“.
6. Care a fost ultimul împrumut înregistrat? Cine a împrumutat și ce carte?	În tabela Operații ne vom poziționa pe ultimul articol și vom afla din tabela Carti titlul și din tabela Cititori numele persoanei.

Structura bazei de date este formată din tabelele Carti1,noperatii și Cititori1,noperatii. Tabelele vor avea structura:

- Carti (cod-carte, titlu, autor, editura, colectia, pret, an-intrare, an-iesire)
- Cititori (cod-cititor, nume, data-nastere, adresa, telefon, ocupația, loc-munca)
- Operații (cod-carte, cod-cititor, Data-impr, Data-rest)



sarcini de laborator

I. Creați structura tabelelor pentru baza de date Bibiloteca.

II. Creați structura unei tabele **FACTURI.DBF** cu date despre facturile neachitate (identificate prin număr, data, valoare) trimise de către furnizori (nume-furnizor, adresa, telefon). Presupunem că la sosirea unei facturi se adaugă o nouă linie, iar la achitare se șterge linia.

Nrfact	data	furnizor	adr	tel	valoare
n,5	d,8	c,10	c,10	n,8	n,8

Dacă dorim să cunoaștem informațiile despre un furnizor, le vom putea obține doar dacă acesta are facturi neachitate. Este firesc? Cum se poate schimba structura astfel încât să eliminăm această anomalie? Cum se numește operația?

1. Introduceți date prin **APPEND**. Adăugați valori fictive în câmpurile Furnizor, adr, tel pe toată lungimea atributelor. Observați trecerea automată la următorul câmp la depășirea lungimii declarate a câmpului curent. Observați avertizarea sonoră.
2. Lansați comanda **SET CONFIRM ON**; reluați adăugarea de valorilor; observați terminarea lungimii unui câmp; se trece automat la următorul câmp?
3. Vrem să introducem mai multe facturi ale aceluiași furnizor, deci numele, adresa și telefonul se vor repeta. Pentru a facilita introducerea datelor, încercați comenzile **SET CARRY TO** și **SET CARRY ON/OFF**.
4. Lansați comanda **SET DATE** cu diferite formate, preluând formatul general din **HELP**. Observați schimbarea datei calendaristice peste tot în tabelă, sub efectul momentan al comenzii.
5. Modificați structura, schimbând tipul câmpului **NRFAC** la șir, iar numele la **NRF**. Afișați datele. Ce se întâmplă cu valorile din primul câmp?
6. Modificați câmpul **FURNIZOR** prin redimensionarea la 4 caractere. Afișați. Modificați lungimea câmpului **VALOARE** prin micșorare la 4 poziții. Afișați.
7. Schimbați ordinea câmpurilor, punând pe prima poziție numele furnizorului și apoi numărul facturii. Afișați. S-au pierdut valori?

III. Evidența judecătorilor din diferite judecătoria și a activității fiecărui judecător este ținută în fișierul **JUDECATORI**. Proiectați structura necesară pentru a afla:

1. Câte cazuri de trafic de valută au fost anul acesta comparativ cu anul trecut?
2. Care au fost dosarele în care verdictul a fost „achitat“?
3. În ce dosare s-a dat pedeapsa maximă?
4. La ce dosare (cazuri) a lucrat persoana X?
5. Câte dintre cazurile la care a lucrat anchetatorul X au fost abandonate?



sarcini pentru realizarea unui mini-proiect

Proiectați structura fiecărei tabele din baza de date specifică aplicației alese de echipaj. Atenție la interogările posibile. Imaginați cereri de informații pentru a folosi toate tipurile de date cu care poate lucra Visual FoxPro.

Vizualizarea, căutarea și sortarea datelor

- Vizualizarea conținutului unei tabele
- Vizualizarea structurii unei tabele
- Căutare secvențială și poziționare în baza de date
- Sortarea și duplicarea unei tabele

Vizualizarea conținutului unei tabele

Afișarea informațiilor dintr-o tabelă se face prin comenzile:

```
LIST / DISPLAY [<lista-expr>] [<domeniu>] [FOR <cond>]
[WHILE<cond>] [TO PRINTER/TO FILE <fis.txt> ] [OFF]
```

În absența oricărei clauze, comanda **LIST** afișează întreaga tabelă, iar comanda **DISPLAY** afișează articolul curent. Clauza **<lista-expr>** enumeră expresiile care vor fi afișate. Clauza **OFF** dezactivează afișarea numărului articolului înaintea primului câmp. Clauza **<domeniu>** poate avea valorile:

Valoare	Efect
ALL	Specifică toate articolele fișierului.
NEXT <n>	Următoarele <n> articole față de articolul curent.
REST	Toate articolele până la sfârșitul fișierului.
RECORD <n>	Selectează doar articolul cu numărul <n>.

Clauza **FOR <cond>** permite selectarea articolelor care îndeplinesc condiția; condiția este verificată pe tot domeniul indicat sau implicit. Clauza **WHILE** permite selectarea articolelor câtă vreme condiția este adevărată. Execuția comenzii încetează la primul articol care nu îndeplinește condiția din **while**.

Clauza **WHILE** presupune mai întâi verificarea condiției, apoi executarea codului (de un număr de ori care poate ajunge chiar și la infinit), iar clauza **FOR** are adesea un contor sau o variabilă de ciclare, ce permite cunoașterea numărului de iterații.

Exemplu

Fie tabela **Studenti** cu informații despre studenții unei facultăți. Vom reține codul, numele, grupa, anul și numele unui curs opțional.

```
use studenti
list    && LIST are domeniul implicit ALL
```

#	cods	numes	grupa	anul	numec
1	1	Dumitru Alina	1	1	info
2	2	Luca Stefan	1	1	mate
3	3	Amarandei Ion	1	2	info
4	4	Alexa Ioana	2	1	engleza
5	5	Nucu Mary	3	2	chineza

Go top && poziționare pe primul articol

List "*", grupa, "*", numes while anul =1 && afișare câtă vreme anul=1

#	*	grupa	*	nume
1	*	1	*	Dumitru Alina
2	*	1	*	Luca Stefan

List "*", grupa, "*", numes for anul =1 &&afișare studenți din anul=1
&& observați răspunsul sistemului

#	*	grupa	*	nume	numec
1	*	1	*	Dumitru Alina	info
2	*	1	*	Luca Stefan	mate
4	*	2	*	Alexa Ioana	engleza

Vizualizarea structurii unei tabele

Vizualizarea structurii unei tabele se realizează prin comenzile:

DISPLAY/LIST STRUCTURE

Comenzile au ca efect afișarea structurii tabelului deschis în mod curent în zona de lucru.

Căutare secvențială și poziționare în baza de date

Căutarea se poate face atât secvențial (pe tabele neordonate), cât și rapid, pe tabele indexate, așa cum vom vedea în lecțiile următoare. Toate comenzile de căutare poziționează pointerul de fișier pe prima apariție a cheii, dacă aceasta este găsită.

Comandă	Efect
LOCATE FOR <cond> [<domeniu>]	Căutare și poziționare pe articolul care îndeplinește condiția <cond>. Căutarea se face într-un domeniu dat (domeniul implicit este ALL).
CONTINUE	Poziționare pe următoarea înregistrare care respectă condiția.
=LOOKUP (<cmp1>, <exp>, <cmp2>)	Returnează valoarea câmpului <cmp1> din primul articol, unde <cmp2> are valoarea egală cu <exp>, altfel șirul vid.
=FOUND ()	Returnează .T. dacă articolul a fost găsit.
=EOF () / BOF ()	Returnează .T. dacă suntem poziționați la sfârșitul tabelului sau înaintea primului articol.
GO TO <n>[In <Z>] GO BOTTOM/TOP	Poziționarea fizică a pointerului fișierului curent (în lipsa clauzei IN) sau a tabelului deschis în zona <Z>. Poziționarea se poate face pe primul articol (TOP), pe articolul cu numărul <n> sau pe ultimul articol (BOTTOM). La deschiderea unei tabele, pointerul se poziționează pe primul articol.
SKIP [+/-]<n>[IN <z>] =RECNO ()	Salt cu avans (+) sau devans (-) în tabelă, peste <n> articole. Numărul articolului curent.

Exemplu

Fie tabela **STUDENT.DBF** cu următorul conținut:

#	cods	numes	grupa	anul	numec
1	1	Dumitru Alina	1	1	info
2	2	Luca Stefan	1	1	mate

3	3	Amarandei Ion	1	2	info
4	4	Alexa Ioana	2	1	engleza
5	5	Nucu Mary	3	2	chineza

Use student	Deschiderea tabelii
? recno()	Poziționare pe pe primul articol
<u>1</u>	
skip -1	Devans cu o poziție
? bof()	
<u>.T.</u>	funcția BOF întoarce true
locate for anul=2	Căutăm un student din grupa 2; poziționarea s-a
? eof ()	facut în interiorul fișierului:
<u>.F.</u>	funcția eof () returnează fals.
display numes_	
<u>Amarandei Ion</u>	Afișăm articolul căutat
continue	
display numes	Căutăm un alt student din grupa 2
<u>Nucu Mary</u>	
continue	
<u>End of locate scope</u>	
? eof ()	Mesajul de căutare fără succes; suntem la sfârșitul
<u>.T.</u>	fișierului? Da
?lookup (numes ,1 ,grupa)	Aflăm numele primului student din grupa 1
<u>Dumitru Alina</u>	
Skip 1	Facem saltul peste articolul găsit
?lookup (numes ,1 ,grupa)	O nouă căutare nu schimbă rezultatul
<u>Dumitru Alina</u>	

Zonele de lucru sunt izolate. Modificarea pointerului de înregistrare ca urmare a unei acțiuni într-o tabelă nu poate determina modificarea pointerului alteia, deschisă în altă zonă de lucru. Fac excepție de la această regulă fișierele înlănțuite cu *SET RELATION*.

Sortarea și duplicarea unei tabele

a. Comanda SORT rearanjează fizic articolele tabelii active, depunându-le într-o altă tabelă, indicată în comandă prin clauza TO.

```
SORT TO <fis.dbf> ON <cheie> [/A] [/D] [/C]
[ASCENDING/ DESCENDING] [FIELDS <lista-camp>]
[<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Criteriul de ordonare este format din una sau mai multe chei. O cheie este un câmp al tabelii. Se pot folosi toate câmpurile, cu excepția celor de tip Memo și General. Pentru fiecare cheie se specifică sensul ordonării: /A=crescător, /D=descrescător. Litera C se folosește pentru a ignora tipul literei. Clauzele ASCENDING și DESCENDING se folosesc la nivelul întregului criteriu de ordonare. Dacă apar atât clauzele locale de indicare a sensului asupra unei chei, cât și cele globale, primele au prioritate.

Atenție

Operația de sortare duplează datele! Este preferabil ca după folosirea rezultatelor sortării să ștergeți fișierele de manevră și să păstrați baza de date inițială.

b. Comanda COPY permite copierea întregului conținut al unei tabele (sau o parte din ea) în altă tabelă.

```
COPY TO <fis.dbf> [FIELDS <lista-câmp>] [STRUCTURE] /
[[<domeniu>] [FOR <conditie>] [WHILE <conditie>]]
```

Articolele tabelii active vor fi copiate într-o nouă tabelă, al cărei nume este precizat în clauza **TO <fis.dbf>**. Clauzele de selecție a câmpurilor (**FIELDS**) și de filtrare **<domeniu>**, **FOR**, **WHILE** permit fixarea structurii și a conținutului noii tabele.

Exemple

1. Dorim afișarea elevilor din Iași pe clase, pentru fiecare clasă în ordinea descrescătoare a mediilor, iar la medii egale în ordine alfabetică.

Considerăm structura Elevi(cod N(5), nume C(10), clasa C(3), adresa M, Media N(5,2)).

```
Use elevi
Sort on cls/ac, media/d, nume/ac to man for "iasi" $ adresa
Use man
List
```

2. Dorim spargerea tabelii Elevi în două tabele, Caminiști și Externi, fiecare din ele având datele ordonate alfabetic.

```
Use elevi
Sort on nume to man
Use man
Copy to caminist for caminist
Copy to externi for not caminist
Use
Erase elevi.dbf
Erase man.dbf
```

3. Reluăm interogările² folosite la identificarea tipurilor de date în baza de date BIBLIOTECA și le rezolvăm. Vom deschide tabele în trei zone:

```
Use carti in 1
Use cititori in 2
Use operatii in 3
```

a) Care sunt cărțile existente (ordonate după autori)?	<pre>sele 1 sort on autori to man use man list titlu, autori, editura, colectia, pret</pre>
b) Ce cărți au intrat sau au ieșit după anul 1994?	<pre>sele carti list titlu, autori for an-intrare>=1994 or; an-iesire>=1994</pre>
c) Care este lista cărților din colecția „De dragoste“?	<pre>sele carti list for colectie="dragoste"</pre>
d) Care este lista cititorilor înscriși care au vârsta sub 15 ani?	<pre>sele 2 list for year(date())-year(data-nastere)<15</pre>

² Revedeți lecția „crearea tabelor“

e) Sunt și profesori de la Liceul de Informatică înscriși la bibliotecă?	<pre> Sele cititori Locate for ocupatie="prof" and; loc-munca="informatica" ? iif (found() "da", "nu") </pre>
f) Sunt restanțieri?	<pre> sele operatii set filter to empty(data-rest) and; date()-data-impr>14 copy to man use man in 4 ? iif (reccount(4)=0,"nu", "da") </pre>
g) Care a fost ultimul împrumut înregistrat? Cine a împrumutat și ce carte?	<pre> sele operatii go bottom sele carti ? lookup(titlu, a.cod-carte, b.cod-carte) sele cititori ?? lookup(nume, a.cod-cititor, c.cod-cititor) </pre>



sarcini de laborator

I. Fie următoarele tabele:

ECHIPE (echipa C(9),grupa C(1))	Conține numele și grupa fiecărei echipe.
JUCATORI (nume C(9), echipa C(9), intrare D, iesire D)	Este un istoric cu echipe și jucători.
CAMPIONAT (e1 C(9), e2 C(9), data D, loc C(9),p1 N(2), p2 N(2), arbitri M)	Păstrează meciurile campionatului.

Sarcini³:

- Afișați echipele în care a jucat „Popescu” și când.
- Afișați arbitrii care au fost la primele 3 meciuri.
- Afișați unde a arbitrat „Popescu”, la ce meciuri, când.
- Afișați echipele din grupa A.
- Afișați componența echipei RAPID (la ultimul joc).
- Afișați localitățile unde a jucat sau va juca echipa Rapid.
- Afișați echipele care au jucat acasă și au pierdut.
- Afișați componența echipelor care joacă la data {01.06.99}, pe stadionul Plăieșilor. Presupunem că este un singur meci.
- Afișați căpitanii celor două echipe care joacă în primul meci înregistrat (presupunem că prima persoana trecută în fișier la o echipă este căpitanul).
- Doi prieteni fotbaliști, Albu și Barbu, se întâlnesc în campionat de două ori (tur și retur). Aflați data și locul.
- Listați programul competițional de săptămâna viitoare ordonat cronologic.
- Afișați meciurile după numele echipei (care joacă acasă), iar pentru fiecare echipă după data desfășurării.

³ Vedeti sugestii de rezolvare la sfârșitul cărții.

II. Organizați datele și scrieți comenzile pentru următoarele sarcini:

1. Lista tuturor persoanelor care locuiesc la adresa X.
2. Care sunt proprietățile unei persoane X date prin buletinul de identitate?
3. Situația imobilelor de pe strada X sub forma următorului raport:

nrc	tip imobil	suprafața	nr apartamente	tip încălzire
	(vilă, bloc, casă)			

4. Care sunt apartamentele unde locuiesc mai mult de două persoane într-o cameră?
5. Câte apartamente sunt proprietate personală în imobilul X?

III. Stabiliți structura bazei de date pentru următoarea situație-problemă!

Compania națională de electricitate dispune de un sistem de evidență a consumului de energie electrică și a achitării sumelor datorate de consumatori.

1. Există lucrători CONEL care citesc periodic contoarele fiecărui abonat și înca-sează sumele corespunzătoare consumului (pentru perioada anterioară citirii).
2. Uneori se modifică prețul unitar pe KW/H la energie electrică pentru consumatorii casnici sau industriali (știm că sunt prețuri diferite!).
3. Înscrierea unor consumatori noi se face pe baza unei cereri de conectare la rețea, ocazie cu care se achiziționează și un contor nou, care este montat și citit de lucrătorii firmei.
4. Scoaterea din evidență se poate face la cererea abonatului sau la deconectarea de la rețea (penalizare din cauza restanțelor).
5. Facturarea înseamnă calculul sumelor de achitat pentru fiecare consumator și expedierea facturilor la consumatori prin angajați speciali.
6. Achitarea unei facturi se face fie direct la angajații CONEL care aduc acasă factura, fie la sediul firmei (caz în care trebuie prezentată factura!).
7. Trebuie avertizate persoanele restante la plata energiei (care au cel mult 2 luni de întârziere). Pentru fiecare lună de întârziere se percepe o taxă!
8. Dacă nu este achitată factura pe 3 luni, are loc deconectarea de la rețeaua de distribuție.

IV. Imaginați activitatea desfășurată de ROMTELECOM! Prin ce diferă de CONEL? Putem avea același model de bază de date? Găsiți și alte domenii de activități similare.



sarcini pentru realizarea unui mini-proiect

Imaginați operații de vizualizare, căutare și sortare a datelor din baza de date specifică domeniului analizat. Scrieți comenzile necesare.

Actualizarea datelor

- Adăugarea articolelor
- Ștergerea articolelor
- Modificarea sau corectarea datelor
- Actualizarea interactivă a tabelelor
- Lucrul cu câmpurile de tip Memo
- Lucrul cu câmpurile de tip General

Adăugarea articolelor

Adăugarea articolelor se face la sfârșitul tabelii active, fie preluând datele în mod interactiv, prin introducerea lor directă de către operator, fie din altă tabelă Fox, fie dintr-un masiv (tablou de memorie), fie chiar din alt tip de fișier (Excel, text etc.). Vom discuta pe rând toate aceste modalități de adăugare a datelor pe parcursul mai multor lecții.

- a. Adăugarea unui articol vid. Un câmp vid are una dintre valorile: zero pentru câmpul numeric, spațiu pentru câmpul caracter, .F. (fals) pentru câmpurile logice, valoarea {} sau { / / } pentru dată calendaristică. Comanda de adăugare este **APPEND BLANK**.
- b. Comanda **APPEND FROM** permite adăugarea la tabela activă a datelor din altă tabelă, specificată în clauza **FROM**.

```
APPEND FROM <fis.dbf> [FOR <cond>] [FIELDS <lista-câmp>]
```

În mod implicit, sunt preluate toate câmpurile. Condiția din clauza **FOR** este testată după plasarea articolului pe noua structură. Clauza **FIELDS** permite selectarea câmpurilor care vor fi folosite la adăugare.

Exemplu

Știind că în fișierul **CONCURS.DBF** (cods, numes, bi, facultate, media, admis) sunt înregistrați participanții la concursul de admitere la facultate, copiați înregistrările celor care au reușit în tabela **STUDENTI** (cods, numes, grupa, anul), existentă deja.

1	10	Dumitru Alina	1	1	info
2	2	Luca Stefan	1	1	mate
3*	5	Andrei Sara	5	2	italo

```
Use studenti
Append from concurs
from admis
```

Eronat. Condiția dată în clauza **FOR** este testată după plasarea articolului pe noua structură, iar tabela Student nu are câmpul **admis**.

```
use concurs
copy to man for admis
use studenti
append from man fields
nume,bi,media
```

Corect. În mod implicit se preiau date din toate câmpurile cu același nume Clauza **FIELDS** permite selectarea câmpurilor care vor fi folosite la adăugare.

Ștergerea articolelor

Operația de ștergere a articolelor dintr-o tabelă activă se realizează în două etape:

Etapa 1. Are loc o ștergere logică sau o *marcare* pentru ștergere, care poate fi ignorată sau nu de comenzile de căutare sau afișare și care poate fi anulată prin comanda **RECALL**.

Etapa 2. Are loc o ștergere fizică efectivă, situație în care datele sunt pierdute definitiv.

a. Comanda DELETE marchează pentru ștergere articolele care îndeplinesc condițiile de filtrare. Comanda acționează pe articolul curent.

```
DELETE [<domeniu>] [FOR<cond>] [WHILE<cond>]
```

Marcarea pentru ștergere nu influențează nici comanda de afișare (observăm „** înaintea primului câmp!), nici o eventuală căutare prin **LOCATE**, o copiere (**COPY**) sau o sortare (**SORT**) etc. Acest lucru se datorează valorii **OFF** pe care este poziționată implicit comanda comutator **SET DELETED ON/OFF**. Setarea comutatorului pe valoarea **ON** determină ignorarea articolelor marcate pentru ștergere.

Pentru a afla dacă un articol este sau nu marcat pentru ștergere se folosește funcția **DELETED** ([<zona>]) care returnează **.T**. dacă articolul curent din zona indicată prin <zona> este marcat pentru ștergere.

b. Comanda PACK permite ștergerea fizică din fișier a tuturor articolelor marcate anterior, fără posibilitatea de recuperare a datelor. Clauza **MEMO** este folosită pentru diminuarea spațiului de disc nefolosit din fișierul Memo asociat, fără a afecta baza de date. Clauza **DBF** este folosită la ștergerea articolelor marcate din baza de date, fără a modifica fișierul Memo asociat.

```
PACK [MEMO] [DBF]
```

c. Comanda ZAP permite ștergerea definitivă din fișier a tuturor articolelor, fără marcarea prealabilă.

d. Comanda RECALL permite revenirea unui articol la starea anterioară operației de ștergere, numai în cazul unei ștergeri logice.

```
RECALL [<domeniu>] [FOR<cond>] [WHILE<cond>]
```

Acțiunea comenzii are articolul curent ca domeniu implicit.

Exemplu

```
use studenti      && implicit comutatorul SET DELETED este OFF
delete record 3   && articolul 3 este marcat pentru ștergere
list
```

<pre>use student</pre>	Deschidem baza de date din exemplul anterior
<pre>delete for cods>=5</pre>	Marcăm articolele cu cods>=5
<pre>recall all for grupa # 1</pre>	Anulăm marcarea celor care au grupa<>1
<pre>pack</pre>	Ștergem efectiv articolele din grupa 1 cu cod>6.

Modificarea sau corectarea datelor

Pentru corectarea valorilor din tabela cea mai recent selectată cu expresii ce pot fi evaluate în momentul executării comenzii se folosește comanda **REPLACE**, cu formatul:

```
REPLACE <câmp1> WITH <exp1> [, <câmp2> WITH <exp2>...]
[domeniu] [FOR<cond>] [WHILE<cond>]
```

Comanda permite înlocuirea valorii existente în câmpul <câmp1> cu valoarea expresiei <exp1>, a valorii existente în <câmp2> cu valoarea <exp2>. Domeniul implicit este articolul curent. Se pot folosi clauzele de filtrare <domeniu>, **FOR**, **WHILE**.

Exemplu

Fie tabela **STUDENT.DBF** cu următorul conținut:

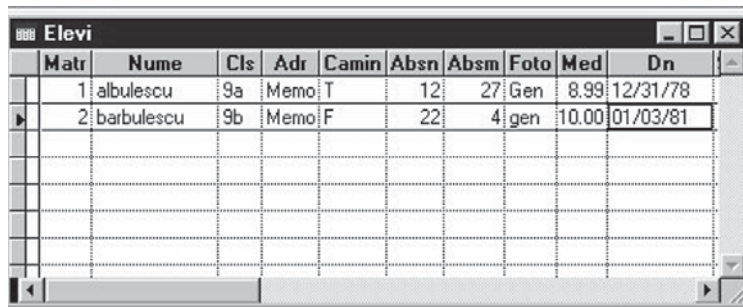
1	11	Dumitru Alina	1	1	info
2	23	Luca Stefan	1	1	mate
3	34	Amarandei Ion	1	2	info

```
use student
repl all cods with recno() && modificăm peste tot codul la numar articol
repl grupa with 4, anul with 5 for numes=[Luca]
list && observați corecția făcută!!
```

1	1	Dumitru Alina	1	1	info
2	2	Luca Stefan	4	5	mate
3	3	Amarandei Ion	1	2	info

Actualizarea interactivă a tabelelor

Comanda BROWSE este una dintre cele mai folosite comenzi pentru vizualizarea și actualizarea datelor. Efectul ei este afișarea tabelului activ pe linii și coloane.



Matr	Nume	Cls	Adr	Camin	Absn	Absm	Foto	Med	Dn
1	albulescu	9a	Memo:T		12	27	Gen	8.99	12/31/78
2	barbulescu	9b	Memo:F		22	4	gen	10.00	01/03/81

Figura 7-1: Tabela afișată prin comanda **Browse**

Pe prima linie sunt afișate denumirile câmpurilor din structura tabelii, iar în continuare sunt liniile cu date.

Formatul comenzii cu cele mai uzuale clauze este:

```
BROWSE [FIELDS <câmp1> [:R] [:V] [<câmp-calc1>=<exp1>]
[LOCK < nr>] [<domeniu>] [FOR <conditie>] [FREEZE <nume-câmp>]
[NOAPPEND] [NOMENU] [NOEDIT] [NODELETE]
```

- Clauza **FIELDS** permite enumerarea câmpurilor care vor forma coloanele tabelului; în lipsa clauzei se rețin toate câmpurile din baza de date, în ordinea structurii. Pentru un câmp putem interzice editarea [:R] sau putem preciza condiția de validare la introducerea valorilor în câmpul respectiv [:V]. În lista de câmpuri pot apărea și câmpuri calculate care primesc un nume și o expresie de calculare. Câmpurile calculate nu pot fi editate, ci numai afișate, dar valorile din ele se modifică odată cu modificările în câmpurile ce formează expresia de calculat.
- Clauzele <domeniu> și **FOR** permit selectarea liniilor care vor fi afișate în fereastră după comanda **Browse**.
- Clauza **LOCK <nr>** permite înghețarea pe ecran a primelor <nr> coloane (câmpuri) în timpul derulării prin **BROWSE** spre stânga sau spre dreapta. Se recomandă ca în structura conceptuală să fie afișate informațiile de identificare a unui obiect la început. Dacă, totuși, acest lucru nu este realizat, putem schimba ordinea de afișare pe ecran a câmpurilor prin clauza **FIELDS**, astfel încât să avem pe primele coloane succesive informațiile necesare.
- Clauza **FREEZE <nume-câmp>** permite menținerea cursorului pe o singură coloană.
- Clauza **NOAPPEND** interzice adăugarea de noi articole în fișier; în lipsa clauzei este posibilă adăugarea.
- **NOMENU** nu afișează linia de meniuri și împiedică accesul la meniuri.
- **NODELETE** împiedică ștergerea accidentală de articole.
- **NOEDIT** interzice editarea articolelor.

Exemplu

Fie fișierul **ELEVI.DBF**(Matr N(3), nume C(10), cls C(3), adr M, camin L, poza G, absm N(2), absn N(2), foto G, med N(5,2)). Sarcini:

Afișați toți elevii, înghețând primele 3 coloane.	Browse lock 3
Afișați numai numele și clasa elevilor, precum și totalul absențelor. Numele nu poate fi modificat!	browse fields cls'nume:R'; absente=absn+absm
Afișați numai elevii din clasa a IX-a A.	browse for cls=" 9a"
Afișați primii 20 de elevi, fixând pentru modificare câmpul Absm.	browse next 20 freeze absm
Afișați elevii clasei a XI-a B, având drept coloane nume_elev, clasa, media.	Browse fields nume:H

Odată cu afișarea ferestrei prin **Browse** pe bara de meniuri apare un submeniu numit **Table**.

Properties deschide o casetă de dialog care permite filtrarea datelor, selectarea câmpurilor, precizarea indexului activ etc.

Go to Record permite poziționarea pe o anumită înregistrare, dată prin numărul ei sau prin căutare cu **Locate**.

Append New Record adăugă o linie nouă.

Toggle Deletion Mark marchează pentru ștergere linia curentă.

Append Records permite compunerea comenzii **APPEND FROM**.

Delete Records deschide caseta de dialog pentru ștergerea articolelor.

Recall Records permite anularea marcajelor de ștergere.

Remove Deleted ștege efectiv articolele marcate.

Replace Field deschide ecranul pentru compunerea comenzii **Rep1ace**.

Size Field permite modificarea interactivă a dimensiunii câmpului pe care este cursorul (dimensiunea se schimbă prin clic).

Move Field permite schimbarea ordinii coloanelor. Se selectează o coloană și se execută tragere și plasare (*drag and drop*) pe noua poziție.

Resize Partitions permite afișarea tabelului în două partiții.

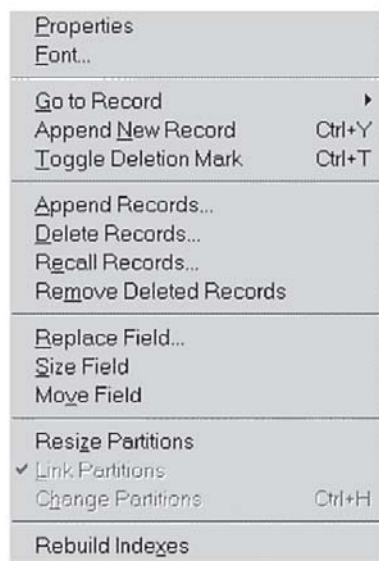


Figura 7-2: Meniul Table

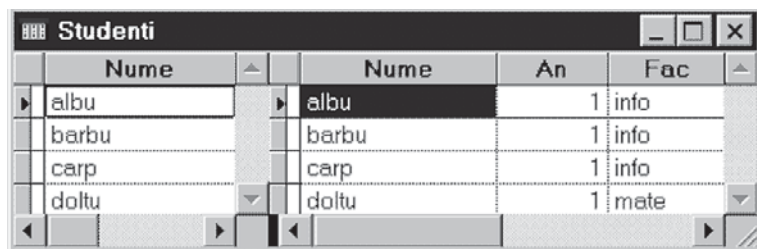


Figura 7-3: Două partiții ale tabelului

Pentru separare, se poziționează cursorul în colțul din stânga jos (colorat cu negru) și se trage partiția până la obținerea dimensiunii dorite. Cele două partiții pot avea o evoluție independentă sau nu (**Link Partitions**).

Studiul de caz conține prezentarea operațiilor de actualizare pentru baza de date analizată:

1. Scoaterea din evidența bibliotecii a unui cititor se face la cererea sa. Se vor șterge și articolele din tabela Operații. Dacă are restanțe, va apărea un mesaj.

<i>Se citește numele cititorului care se retrage.</i>	<code>Accept 'nume cititor' to x</code>
<i>Se caută în tabela CITITORI și se reține codul.</i>	<code>Sele cititori Cod=lookup(cod_cit, x, nume) Delete all for nume=X Pack</code>
<i>Din tabela OPERAȚII se copiază într-o singură operație toate articolele cititorului care au data_restituirii necompletată, pentru a vedea dacă are restanțe.</i>	<code>Sele operatii Set filter to cod_cit=y Copy to man2 for empty(data_rest) Delete all Pack</code>
<i>Este afișat un mesaj dacă sunt cărți nerestituite.</i>	<code>Use man2 in 4 ? IIf(reccount(4) #0,"sunt restante","")</code>

2. Împrumutarea unei cărți:

<i>Se citește codul cărții.</i>	<code>Input' cod carte=" to codx</code>
<i>Se citește codul cititorului.</i>	<code>Input 'cod cititor=?' to cody</code>
<i>Se adaugă o nouă operație de împrumut.</i>	<code>Select operatii Append blank Replace cod_carte with codx, cod_cit with cody, data_impr with date()</code>

3. Restituirea unei cărți:

<i>Se citește codul cărții și al cititorului.</i>	<code>Input "cod carte" to x Input "cod-cititor" to y</code>
<i>Se caută articolul și se completează data restituirii. Presupunem că datele sunt corecte, altfel ar trebui folosită structura alternativă, pe care o vom învăța mai târziu!</i>	<code>Select operatii Locate for cod_carte=x and cod_cit=Y and data_rest={//} replace data_rest with date()</code>

4. Înscirerea unui cititor se face pe baza solicitării acestuia:

<i>Se citește codul cititorului, numele, adresa.</i>	<code>accept "cod-cititor" to X Accept "nume? " to y Accept "adresa?" to Z Accept "data nasterii?" to w</code>
--	--

Se caută articolul și se completează data restituirii. Presupunem că datele sunt corecte, altfel ar trebui folosită structura alternativă, pe care o vom învăța mai târziu!

```
Select cititori
Append blank
replace cod_cit with val(x)
replace nume with y, adresa with y
replace data_n with ctod(w)
```

5. La un control s-au constatat niște nereguli. Urmăriți în prima coloană aceste nereguli, iar în cea de-a doua acțiunile de corectare.

a) Există cărți vechi, deteriorate sau pierdute, care ar trebui trecute în altă evidență!

```
Select carti
delete for uzat
copy to deterior for deleted()
pack
```

b) Nu a fost aplicată majorarea prețului cu 25% la toate cărțile intrate în bibliotecă înainte de anul 1990!

```
replace all pret with pret*125 for
year(dataintr)<=1990
```

c) Editura „ALBATROS” și-a schimbat numele în „PINGUIN” la 1 septembrie 1992. Nu s-a făcut această modificare!

```
set date to german
replace editura with 'PINGIUN' for
editura= 'ALBATROS' and
dataintr>={01.09.92}
```

d) Numele autorului G. B. SHAW a fost scris greșit, sub forma „G.B.SOU”.

```
repl autor with 'G. B. SHAW' for
autor= 'G.B.SOU'
```

e) Cărțile autorului „POPA ION” ar trebui marcate, întrucât s-a dovedit că este un plagiator, chiar dacă numai cărțile apărute după revoluție, vor fi scoase definitiv din evidență!

```
delete all for upper(autor)="POPA ION"
recall for dataintr<={22.12.89}
pack
```

f) De ce sunt manualele trecute în fișierul „Manuale.dbf” dacă are aceeași structură? Concatenați fișierele de cărți și de manuale!

```
Use manuale
Replace all alte_inf with "manual"
Use carti
Append from manuale
Use
Delete file manuale.dbf
```

g) Nu au fost listate separat manualele!

```
Use carti
List for alte_inf='manual' to print
```

h) Mutați cartea de cod =200 sfârșitul tablei Carti.
Vom scrie un program lansând editorul de programe cu **Modify Command**.
Vom apela prin **DO <nume>**

```
Var 1.prg
use carti
copy to man for
cod_carte=200
pack
append from man
erase man.dbf
```

```
Var 2.prg
use carti in 1
use carti in 2 again
sele 1
locate for cod_carte=200
sele 2
append blank
replace b.titlu with a.titlu
replace b.autor with a.autor
...etc se vor copia toate campurile
sele 2
use
sele 1
delete
pack
```




sarcini de laborator

I. Sarcini de actualizare interactivă a tabelelor prin comanda Browse

1. Vizualizați prin ecranul Browse tabelele bazei de date BIBLIOTECA
2. Executați următoarele manevre:
 - a) adăugați un articol; poziționați-vă pe articolul 3; ștergeți articolul 4;
 - b) modificați prețul tuturor cărților editurii Ciunafaiul prin majorare cu 10%;
 - c) modificați interactiv numai prețul cărților apărute după 1 decembrie 1998;
 - d) poziționați-vă pe primul articol și afișați titlul și autorul tuturor cărților;
 - e) ștergeți toate cărțile autorului „D.R. Popescu“;
 - f) afișați în fereastra Browse numai cititorii elevi în clasa a 9-a;
 - g) marcați pentru ștergere primii doi cititori;
 - h) anulați marcajul pentru primul; ștergeți fizic al doilea cititor;
 - i) afișați în două partiții tabela Carti; deplasați cursorul pe bara de derulare orizontală în cadrul fiecărei partiții. Se vizualizează câmpuri aparținând acelorași înregistrări sau puteți avea o deplasare independentă? Cum?

II. Știind că mai sunt manuale școlare în tabela LICEU (Cod, titlu, clasa, autor, editura, pret) adăugați articolele în tabela CARTI.DBF (inventar, titlu, autor, preț, editura, data-intrării, data-ieșirii, uzat, alte_inf). Titlul cărții în tabela CARTI va conține și clasa pentru care este manualul (de exemplu, „Informatică clasa a IX-a“).

LICEU.DBF

Informatica	IX	T.Sorin	L&S infomat	57000
Informatica	X	M.Cerchez	Polirom	45000

Care dintre cele două variante rezolvă problema?

Varianta 1.

```
use liceu
copy to man for cls='IX'
use man
replace all titlu;
  with alltrim(titlu)+ "cls."+ cls
use carti
append from man
erase man.dbf
```

Varianta 2.

```
use liceu
delete for between(cls, 9, 12)
repl titlu with;
  titlu+cls for deleted()
use carti
append from manuale;
  for deleted()
```

III. Fie un fișier PERSONAL ținut pentru o societate, cu următoarele informații (cod, nume, număr-buletin, salariu, data-nașterii, nr_copii, alocația, costul-ora-noapte, număr-noapți, suma-spor-noapte, rețineri, rest-plată). Se cere:

1. Afișați persoanele care lucrează în societate.
2. Ce salariu are o persoană al cărei nume este introdus de la tastatură?
3. Care sunt persoanele fără copii în funcții de administrație?
4. Alocația de stat pentru copii se calculează ca o sumă fixă înmulțită cu numărul de copii. Treceți în baza de date alocația.

5. Modificați prin majorare cu 15% salariul lucrătorilor în funcții de contabilitate și cu 30% salariul celor în funcții de administrație.
6. Treceți codul unei persoane ca fiind chiar numărul articolului curent.
7. Afișați pentru fiecare persoană suma de primit, știind că reținerile se scad, iar sporurile și alocația se adună la salariu.
8. Treceți sporul de noapte ca fiind calculat după formula:
spor noapte=salariu mediu/zi*nr. nopți*0.17.
9. Afișați persoanele care au rest de plată <=0.
10. Toate persoanele care lucrează în funcții de administrație, au copii și au peste 60 ani, se vor pensiona (marcare pentru ștergere).
11. Afișați posturile care vor fi vacante după pensionarea personalului.
12. Afișați numele persoanei care are codul 4. Înaintea persoanei (cu codul 4) inserați un articol vid, apoi completați cu date.
13. Completați același salariu cu al primei persoane la toate celelalte care au aceeași funcție.
14. Presupunând că seria buletinului de identitate este pe primele două caractere, să se afișeze toate persoanele care au seria „DK”.
15. Schimbați datele persoanei cu buletinul X cu cele ale persoanei Y.
16. Afișați sub forma următoare datele din baza de date:

BI	nume	funcție	rest-plată	semnătura
----	------	---------	------------	-----------

Lucrul cu câmpurile de tip Memo

Câmpul de tip Memo este necesar – în principal – pentru situațiile când trebuie memorate cantități variabile de informații. De exemplu, pentru înregistrarea studiilor fiecărei persoane, zona respectivă va conține șiruri de lungime variabilă.

PERSONAL.DBF

nume	studii
POPA	ADR1
IONEL	ADR2

PERSONAL.FPT

Academia de Studii Economice Facultatea de Cibernetică, București
Liceul de Informatică, Iași

Dimensionarea câmpului la valoarea maximă (255 de caractere) nu rezolvă întotdeauna problema (s-ar putea ca pentru unele persoane să fie și în acest caz insuficient spațiu, iar spațiul pentru altele să rămână în mare parte nefolosit). Structura devine ineficientă prin mărirea ei. A apărut necesitatea depunerii informațiilor propriu-zise (despre studii!) în alt fișier, sub forma blocurilor de text, făcând legătura cu articolul care ar trebui să le conțină. Fișierul asociat poartă același nume ca și baza de date, are extensia .fpt și se deschide simultan cu aceasta.

Comenzi și funcții utile

Comanda	Efect
MODIFY MEMO <lista-cmp>	Deschiderea unei ferestre de editare pentru fiecare câmp Memo dintr-o listă dată.

Comanda	Efect
APPEND MEMO <memo> FROM <fisier> [OVERWRITE]	Introducerea datelor dintr-un fișier text în câmpul Memo. Clauza Overwrite este necesară când dorim suprascrierea conținutului fișierului text peste vechiul conținut al câmpului Memo. Operația implicită este de adăugare.
COPY MEMO <memo> TO <fis. txt> [ADDITIVE]	Extragerea dintr-un câmp Memo a informațiilor într-un fișier text. Operația implicită este de suprascriere. Dacă dorim ca vechiul conținut să nu se piardă, trebuie folosită clauza ADDITIVE .

Exemple

1. Fie tabela **STUDENT.DBF**

#	cods	numes	grupa	anul	alte_inf
1	1	Dumitru Alina	1	1	Memo
2	2	Luca Stefan	1	1	Memo
3	3	Amarandei Ion	1	2	Memo

Vrem să afișăm informațiile din câmpul **alte_inf** pentru persoanele din "Iasi".
Observați plasarea numelui de câmp Memo în comanda **List**!

```
use student
list alte_inf for "Iasi"$alte_inf
```

```
Record# ALTE_INF
1      adresa: Iasi, str. Vasile Alecsandri nr 13
      telefon: 245678
      buletin: seria AD, număr 457890

3      adresa: Iasi, str Tudor Vladimirescu, nr 5
      bl e4, sc a, et 5, ap 9
      telefon: 672134
      buletin: seria D4, număr 9080897
```

Observați lipsa datelor din articolul 2?

2. În baza de date **TELEFON.DBF** (nume, telefon, adresă) se găsește numărul de telefon al lui „Dumitru Alina“. Să se adauge numărul la conținutul câmpului **alte_inf** din tabela Student.

Varianta 1

```
use telefon
locate for name="Dumitru Alina"
display "tel."+str(telefon);
to file telef. txt
use student
locate for numes="Dumitru Alina"
append memo alte_inf;
from telef. txt
```

Varianta 2

```
use telefon
nrtel=lookup(telefon,;
"Dumitru Alina", nume)
use student
repl alte_inf;
with adresa +str(nrtel);
for numes="Dumitru Alina"
```

3. Să presupunem că la introducerea datelor în fișierul **STUDENT.DBF** s-a inversat conținutul câmpului **alte_inf** pentru „Amarandei“ cu cel a lui „Luca“. Vom corecta această eroare.

```
use elevi
locate for nume="Amarandei"
copy memo alte_inf to man1
x=recno()
locate for nume="Luca"

copy memo alte_inf to man2
append memo alte_inf ;
from man1 overwrite
goto x
append memo alte_inf;
from man2 overwrite
```

4. Folosirea câmpului Memo pentru prelucrarea unui fișier de tip text:

Să presupunem că avem fișierul **Date_pers.txt** și dorim modificarea peste tot a vechiului număr de buletin 'dk123456' cu noul număr 'as121212'

Rezolvare Vom crea o tabelă fictivă cu un câmp Memo în care vom copia fișierul text. Folosim funcția **Strtran** care caută toate aparițiile unui șir – parametru 2 – și le înlocuiește cu alt șir – parametru 3.

Observați programul următor:

```
Create table x(cmp M)
```

```
Append blank
```

```
Append memo cmp from date_pers.txt
```

```
Replace cmp with strtran(cmp, [dk123456], [AS121212])
```

```
Copy memo cmp to date_pers.txt
```

```
Use
```

```
Erase x.dbf
```

Lucrul cu câmpurile de tip General

OLE (Object Linking and Embedding) – legare și încorporare a obiectelor) este un mecanism de comunicare între aplicații prin intermediul căruia o aplicație (numită aplicație client) poate folosi servicii oferite de altă aplicație (numită aplicație server). De exemplu, aplicația Paintbrush este o aplicație server care creează și editează imagini (fișiere **.bmp**) care pot fi manipulate de aplicația Microsoft Word în textele (fișierele **.doc**) proprii.

În FoxPro sub Windows pot fi stocate și manipulate obiecte cum ar fi grafice, texte, imagini, tabele, sunete etc. în câmpurile de tip General. Obiectele, create și/sau editate de alte aplicații Windows pot fi păstrate în câmpul General prin *legare* sau prin *încorporare*. În ambele cazuri, între baza de date și aplicația sursă, cea cu care a fost creat obiectul, se stabilește o legătură care permite lansarea direct din FoxProW a aplicației în momentul în care se dorește editarea obiectului.

1. **Introducerea** unui obiect în câmpul General se face prin comanda:

```
APPEND GENERAL <câmp> FROM <fis>[LINK] [CLASS<id_server>]
```

Comanda *încorporează* în câmpul General **<câmp>** al înregistrării curente o copie a fișierului **<fis>**. Folosind clauza **LINK** fișierul va fi *legat* la baza de date. Trebuie specificat numele complet al fișierului **<fis>** folosit. De obicei, extensia **.XLS** este asociată foilor de calcul din Excel, extensia **.DOC** documentelor Microsoft Word etc. Dacă din numele fișierului nu poate fi determinată aplicația server cu care a fost creat fișierul, atunci trebuie specificat parametrul **CLASS**. Identificatorii aplicațiilor server utilizate frecvent sunt dați în următorul tabel.

Server	Id-server
Microsoft Excel Chart – grafic	ExcelChart
Microsoft Excel – foaie de calcul	ExcelWorksheet
Microsoft Graph – grafic	MSGraph
Microsoft Word 6.0 – document	WordDocument.6
Microsoft Word 6.0 – imagine	Word.Picture.6
PaintBrush – imagine	Pbrush
imagine Bitmap	StaticDib
Sunet	SoundRec
Texte	Textfile

Exemplu: În fișierul CONCURS reținem informațiile despre participanții la un concurs. Câmpul Poza(G, 10) conține fotografia candidatului.

Dacă am scanat poza și am editat-o în aplicația Paint Shop Pro obținând fișierul Pozamea.jpg, comanda de introducere în câmpul General este următoarea:

```
APPEND GENERAL poza FROM 'C:\windows\pozamea.jpg' LINK
```

2. Completarea interactivă a unui câmp General cu un obiect:

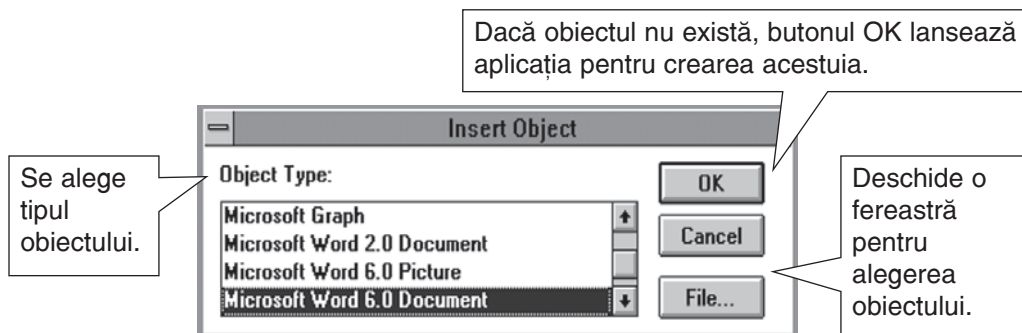


Figura 7-4: Introducerea interactivă a unui obiect într-un câmp General

1. Deschideți fereastra de editare cu combinația **<Ctrl>+<Home>**.
2. Selectați **Edit, Insert Object**.
3. Alegeți tipul obiectului de inserat.
4. Alegeți fișierul din fereastra deschisă prin butonul **File...**
5. Dacă obiectul de inserat nu există, apăsați (executați clic) pe butonul **OK**, care lansează aplicația pentru crearea fișierului.
6. Salvați folosind combinația **<Ctrl>+<W>**.

3. Inserarea obiectelor în câmpul General prin Clipboard

Dacă este necesară păstrarea unei porțiuni dintr-un obiect, puteți folosi **Clipboard-ul** și opțiunea *Paste special* din meniul **Edit**.

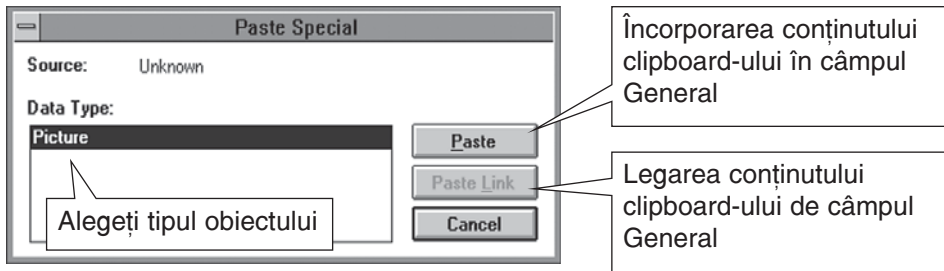


Figura 7-5: Folosirea opțiunii Paste special

4. Pentru **editarea interactivă a unui câmp General** procedați astfel:

a) Poziționați-vă pe câmpul General și deschideți fereastra de editare.

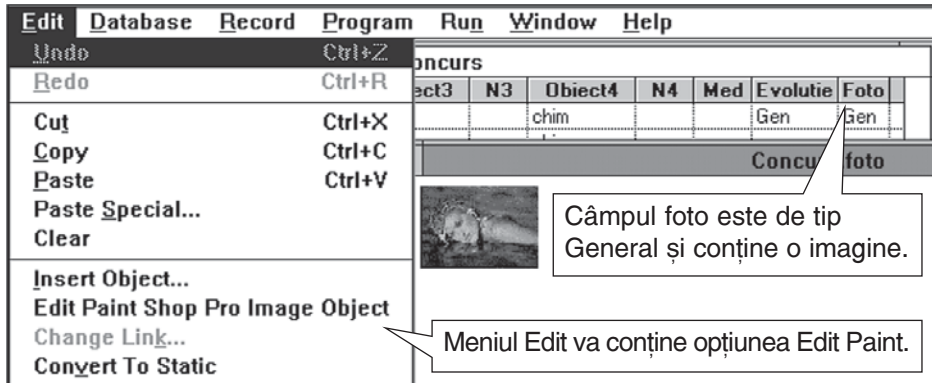


Figura 7-6: Submeniul Edit

b) Deschideți submeniul Edit. Clauza de apelare a aplicației server diferă după tipul obiectului existent în câmp. Observați figura în care am surprins ecranul Browse unde, pentru câmpul foto, s-a deschis fereastra de editare care conține o imagine. Meniul Edit are o opțiune corespunzătoare tipului obiectului (Edit Paint Shop Pro Image Object).

5. **Revenirea în mediul Fox:** după efectuarea corecțiilor în aplicația server, reveniți în mediul FoxPro.

Folosirea câmpului General

Exemple

Dorim să introducem într-o tabelă numele și conținutul mai multor documente pe care le avem într-un director.

Comenzi

```
CREATE TABLE Fisier( nume c(24), text g)
CD GETDIR ()

Nr = ADIR(A, "*.doc")
IF nr > 0
  FOR i = 1 to nr
    APPEND BLANK
  REPLACE fisier.Nume WITH A(i,1)
```

Efect

Crearea tabelii
 Poziționare pe director prin fereastra de dialog
 NR=Număr de fișiere copiate
 Dacă avem documente, atunci adăugăm câte un articol în fișier reținând numele documentului și conținutul său.

```

APPEND GENERAL fisier.text FROM A(i,1)
ENDFOR
ELSE
MESSAGEBOX("Nu sunt documente !")
ENDIF

```

Sarcină. Încercați operația folosind un câmp de tip Memo!



sarcini de laborator

Creați fișierul **Concurs** cu informații despre concurenții la Olimpiada de Informatică, cum ar fi: numele concurentului, școala și localitatea de unde vine, o scurtă prezentare a concurentului (curriculum vitae, fotografia, amprenta vocală), rezultatele la diverse probe.

1. Populați cu articole prin **Browse**.
2. Puneți la primul articol în câmpul Foto o imagine .gif (.bitmap etc.).
3. Puneți la primul articol în câmpul Evoluție un grafic al evoluției rezultatelor respectivului concurent realizat prin Excel (Microsoft) Graph.
4. Puneți la primul articol în câmpul Voce un fișier sunet.
5. Puneți la primul articol în câmpul CV (curriculum vitae) un fișier document Word.
6. Afișați câmpurile speciale.
7. Scrieți un program care să afișeze poza unui concurent dat prin nume de la tastatură.
8. Editați diversele obiecte.
9. Copiați de la articolul unu la articolul doi conținutul câmpului Test.
10. Aflați numele și școala de unde vine primul concurent înscris.
11. Presupunând că în câmpul Total_punctaj se adună rezultatele la cele 3 probe de concurs, realizați calculul prin comenzi adecvate!
12. Afișați concurenții în ordinea descrescătoare a punctajelor, la punctaje egale, alfabetic.
13. Aflați câți concurenți au venit din localitatea X?


```

&& accept 'loc?' to x
&& copy to man for localitate=X
&& use man
&& ? reccount()

```
14. Aflați care este concurentul cu cel mai mare punctaj la proba 1?


```

&& Use concurenti
&& Sort on probal/d to man
&& Use man
&& X=probal
&& List while probal=

```



sarcini pentru realizarea unui mini-proiect

Identificați operațiile de actualizare a datelor din fiecare proiect. Realizați interactiv aceste operații. Scrieți comenzile pentru actualizare.

- Indexarea tabelelor
- Căutarea rapidă și poziționarea în tabela indexată

Indexarea tabelelor

Indexarea este o metodă de accesare rapidă, într-o manieră ordonată, a conținutului unei tabele, fără a duplica datele propriu-zise, fără a le depune în altă tabelă și a avea grijă de ele. Alte avantaje ale indexării sunt posibilitatea folosirii expresiilor drept criteriu de ordonare, selectarea după criterii unice, actualizarea în timp real etc.

Să luăm ca exemplu tabela ELEVI. Parcurgerea ei ordonată după nume se poate face prin construirea unui index care să rețină numărul înregistrării din tabelă și criteriul „nume”. Liniile acestui index sunt ordonate după valorile cheii. Solicitarea de a accesa tabela ELEVI prin cheia „nume” va impune parcurgerea indexului. Pentru fiecare linie din index se ajunge prin pointer la articolul cu datele propriu-zise.

Indexarea realizează o legătură logică între index și fișierul de date prin numărul articolului. Observați desenul următor.

#	nume	pren	cls	absn
1	popescu	teo	11b	5
2	albuleț	geo	11a	4
3	stan	ana	11c	6
4	marinică	ion	11d	0
5	albuleț	ina	9c	7

#	cheie
5	9c
2	11a
1	11b
3	11c
4	11d

#	cheie
2	albuleț
5	albuleț
4	marinică
1	popescu
3	stan

Indecșii pot fi depuși în fișiere **index (.CDX)** asociate tablei, au același nume și se deschid sau se închid odată cu tabela, orice operație de actualizare asupra acesteia reflectându-se automat și asupra tuturor indecșilor membri.

O tabelă poate avea mai mulți indecși, dar numai unul este la un moment dat activ și determină criteriul de parcurgere.

Tipuri de indecși

- **Indexul obișnuit (regular)** este folosit pentru selectarea ordinii de parcurgere din câmpurile non-cheie.
- **Indexul unic (unique)** este folosit pentru selectarea ordinii de parcurgere bazate pe prima apariție a valorii în câmpul specificat.
- **Indexul candidat (candidate)** este folosit în tabelele care au deja fixat indexul primar, dar în care este necesară verificarea valorilor unice și în alt câmp decât în câmpul cheie.

- **Indexul primar (primary)** este folosit în contextul unei *tabele incluse într-o bază de date* și asigură introducerea valorilor unice pentru cheia articolelor. O tabelă poate avea un singur index primar.

Comenzi și funcții utile

1. Pentru crearea unui index este folosită comanda:

```
INDEX ON <exp> TAG <tag> [UNIQUE]
[DESCENDING/ASCENDING] [FOR<cond>]
```

Comanda **Index** permite crearea unui index cu numele **<tag>** pe baza expresiei **<exp>**. Clauza **Unique** specifică tipul indexului. Sensul ordonării este dat de clauza **DESCENDING/ASCENDING**, iar filtrarea este posibilă prin condiția **FOR**.

2. Pentru specificarea numelui indexului activ este folosită comanda:

```
SET ORDER TO TAG <tag>
```

sau comanda

```
USE <bd> ORDER <tag>
```

3. Pentru ștergerea unui index este folosită comanda:

```
DELETE TAG <tag>
```

4. Funcții

Funcție	Efect
=TAG (<nr-index>)	Numele unui reper-index care ocupă poziția <nr index>
=TAGNO (<nume-reper>)	Poziția pe care o ocupă un index dat.
=KEY (<nr-index>)	Expresia cheii de indexare pentru reperul-index care ocupă poziția <nr-index>
=ORDER ([<nr-zona>])	Numele indexului activ.
=TAGCOUNT ([<zona>])	Numărul de repere index din multiindex.

Exemple

1. Fie baza de date ELEVI.
Observați comenzile și efectul lor:

nume	pren	Cls	absn	admis	Dn
popescu	teo	11b	5	.T.	11/11/94
albuleț	geo	11a	4	.T.	09/09/95
stan	ana	11c	6	.T.	07/11/95
marinică	ion	11a	0	.F.	08/09/95
albuleț	ina	11a	7	.F.	09/09/95

```
index on cls tag cls      creare tag cls în fișierul structural elevi.cdx
unique
```

```
list                      acces permis la primul articol al fiecărei clase
```

2	albuleț	geo	11a	4	.T.	09/09/95
1	popescu	teo	11b	5	.T.	11/11/94
3	stan	ana	11c	6	.T.	07/11/95

<code>index on dtos(dn)+nume</code> <code>tag dn</code>	crearea reperului <code>Dn</code> în <code>elevi.cdx</code> permite ordonare cronologică și alfabetică
<code>index on nume + str(abssn) tag x</code>	crearea reperului <code>nume</code> în <code>elevi.cdx</code> permite ordonare alfabetică și după absențe!
<code>index on nume tag nume</code>	crearea reperului <code>nume</code> pentru parcurgerea alfabetică
<code>use elevi order nume</code> <code>sau set order to tag nume</code>	activarea tag-ului <code>nume</code> din <code>Elevi.cdx</code>
<code>set order to 0</code>	se consideră tabela neordonată
<code>delete tag nume</code>	ștergerea indexului <code>nume</code>

2. Scrieți o secvență care permite precizarea de către operator a câmpului, apoi indexarea tabelului ELEV1.

```
accept ' care este câmpul ? ' to cmp
use elevi
index on &cmp tag &cmp
list nume, cls, dn, ob1, trim1
return
```

Atenție la macrosubstituție!

Definirea vizuală a indecșilor

Realizarea interactivă a indecșilor pentru o tabelă presupune deschiderea ferestrei Table Designer. Dacă expresia de indexare conține doar un câmp, atunci indexul se poate defini în primul tab al ferestrei Table Designer (Fields), odată cu descrierea câmpului. Dacă expresia de indexare este compusă din mai mulți termeni (câmpuri, constante, variabile), atunci folosim tab-ul Indexes. Expresia de indexare, ca și expresia de filtrare, pot fi construite cu ajutorul utilitarului Expression Builder.

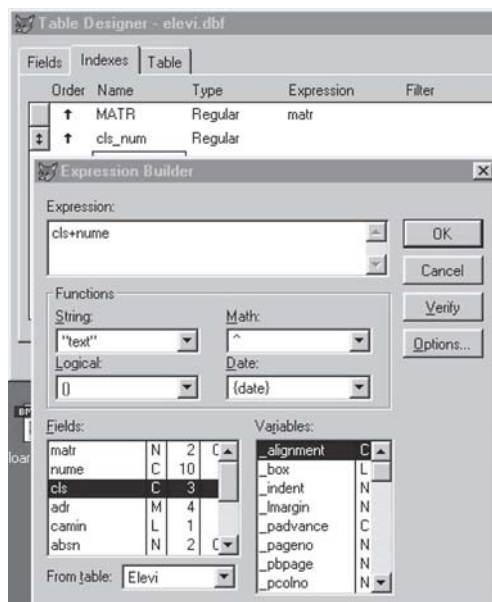


Figura 8-1: Definirea vizuală a indecșilor

Căutarea rapidă și poziționarea în tabela indexată

Una dintre funcțiile importante ale unui SGBD este accesarea rapidă a tabelii. Condiția impusă este ca tabela să fie indexată după expresia de căutare. Există două comenzi de căutare rapidă: **FIND** și **SEEK**; prima folosește drept expresie de căutare o constantă. Dacă dorim căutarea printr-o variabilă va trebui să folosim fie **FIND** cu macrosubstituție, fie **SEEK**.

Căutarea se încheie la primul articol din tabelă care are cheia de indexare egală cu valoarea expresiei, dacă o astfel de înregistrare există. În cazul în care căutarea nu a avut succes, se mută pointerul de fișier pe **EOF** (dacă valoarea comutatorului **SET NEAR** este **OFF**) sau se mută indicatorul pe prima înregistrare care ar fi conținut valoarea dacă ar fi existat (dacă **SET NEAR** este **ON**).

Comenzi și funcții utile

FIND <expC> SEEK <exp>	Comanda de căutare rapidă
=SEEK (<exp>)	Funcție care caută <exp> și returnează .T. dacă a găsit un articol.
= FOUND () / =EOF ()	Funcții care returnează .T. dacă articolul a fost găsit.

Exemple

Fie tabela ELEVI:

1	albuleț	geo	11a	4	.T.	09/09/95
2	albuleț	ina	11a	7	.F.	09/09/95
3	marinică	ion	11a	0	.F.	08/09/95
4	popescu	teo	11b	5	.T.	11/11/94
5	stan	ana	12c	6	.T.	07/11/95
6	galeriu	sanda	9a	0	.F.	/ /

1. Efectele comenzilor Find și Seek:

use elevi order nume find popescu ? found()	Caută persoana cu numele "popescu" în tabela indexată după cheia nume. Șirul nu trebuie pus între delimitatori.
.T.	Funcția FOUND returnează .T.
var='popescu' find &var ? recno() 4	Dacă dorim căutarea cu o variabilă, atunci trebuie să folosim macrosubstituția. Afișăm numărul articolului curent.
index on cls tag cls	Indexăm după codul clasei.
find 9a ? eof() .T. ? found() .F.	Vrem să ne poziționăm pe primul elev din clasa 9a. Căutare eșuată. Observăm funcțiile EOF (care returnează True) și FOUND (care returnează False) pentru eșecul căutării.

<code>find " 9a"</code> <code>? eof()</code> <code>.F.</code>	Valoarea cheii de căutare începe cu spațiu, deci va trebui pusă între ghilimele. A fost găsit un articol, deci poziționarea nu este pe <code>eof</code> .
<code>Set order to tag nume</code> <code>seek var</code> <code>? found()</code> <code>.T.</code>	Inițializarea variabilei s-a făcut anterior cu valoarea Popescu. Căutarea prin <code>seek</code> conduce la același rezultat ca și folosirea comenzii <code>FIND</code> .
<code>? seek("popescu")</code> <code>.T.</code>	Observați funcția de căutare, care întoarce valoarea adevărat sau fals.

2. Fie baza de date MECIURI care reține toate meciurile unui campionat:

#	cod	e1	e2	loc	data	ora
1		dinamo	farul	constanța	08.09.95	10
2		poli	corvinul	iași	08.09.95	15
3		rapid	dinamo	bucurești	07.09.95	10
4		steaua	rapid	bucurești	08.09.95	15

Să se completeze *codul* fiecărui meci cu o informație care să identifice poziția meciului în programul competițional, cronologic vorbind, pentru aceeași dată în ordinea alfabetică a locului de desfășurare și, dacă sunt meciuri în aceeași zi, în același loc, le vom aranja în funcție de oră.

Varianta 1

```
use meciuri
sort on data, loc, ora to man
use man
repl all cod with recno()
use
erase meciuri. dbf
rename man. dbf to meciuri.
dbf
```

Varianta 2

```
use meciuri
index on
dtos(data)+loc+str(ora; tag v
I=1
scan
repl cod with i
i=i+1
endscan
use
```



sarcini de laborator

1. Fie o tabelă **CINEMA** (cod-film, nume-film, producător, regia, anul, gen, scenariu, premii) cu principalele producții cinematografice din anul 1948 încoace și o tabelă **ACTORI** (nume, data-nașterii, sex, cod-film, naționalitate, data-deces) cu informații despre stelele filmului. Se cere:

- codificarea filmelor, odată cu înregistrarea lor, printr-o informație compusă: nume-țară + „-“ + genul + „-“ + număr-curent;
- afișați care a fost primul film (cronologic) al fiecărui gen;
- scrieți țările ale căror producții au fost înregistrate în baza de date.
- afișați filmele produse în UZBEKISTAN în anul 1994;
- lista filmelor pe genuri;
- afișați scenariștii pe genuri cinematografice, după 1990;

- g) ce actori au jucat în filmul „Pe aripile vântului“?
- h) cu cine a mai jucat actorul Ralph Macchio în „Karate Kid“?
- i) lista actrițelor în viață care n-au depășit 30 ani, pe naționalități;
- j) primii 5 „veterani“ ai filmului românesc.

2. Verificați dacă cele două secvențe produc același rezultat!

```
Secvența 1:
use elevi
index on cls tag cls
find '11a'
? iif (found(), 'da', 'nu')
```

```
Secvența 2:
use elevi
index on cls tag cls
? iif (seek
('11a', 'cls'). 'da', 'nu')
```

3. Dorim să găsim media elevului Stan Ana din 12C folosind comanda **Seek**. Care variantă este corectă?

```
use elevi
index on cls+nume+pren tag x
seek "12CStan Ana"
? iif(found(), media, 'nu este eleva')
```

```
Use elevi
Index on cls+nume+pren tag x
Seek " Stan Ana 12c"
? iif(not eof(), media, 'nu este
eleva')
```

4. NERO, împăratul roman, îi acuză pe creștini de incendierea Romei și se hotărăște să-i pedepsească.

- a) Cere să se facă o listă a tuturor persoanelor din ROMA cu numele, vârsta și dacă este sau nu creștin.
- b) Pe toate fetele sub 25 ani le expulzează din ROMA.
- c) Pe toți copiii între 3 și 10 ani a căror nume începe cu „A“ îi iartă.
- d) Hotărăște să ierte pe cel mai bătrân om din ROMA, fie el femeie sau bărbat.
- e) Femeile căsătorite cu vârsta între 25 și 40 ani sunt și ele iertate, cu condiția să aducă ofrande zeiței DIANA. Se bucură în sinea lui când află că numai o singură femeie, MARIA, a refuzat oferta.
- f) Hotărăște ca toți ceilalți să fie omorâți, dar lista acestora o pune pe monumentul funerar „in memoriam“.
- g) Apoi împăcat, NERO dă poruncă să fie aduse înapoi fetele expulzate și scoate o „foaie“ cu cetățenii „de bună credință“ ai Romei.

Traduceți acțiunile lui NERO în limbajul FoxPro.

5. Fie date mai multe cărți de joc identificate prin culoare (treflă, caro, cupă, pică) și valoare (2-14). Asul este 11. Verificați dacă pachetul este complet!

- 6.** Pentru fișierul ELEVI (nume, clasa, media), să se afle:
- a. numele elevilor cu cea mai mare medie din fiecare clasă;
 - b. lista claselor;
 - c. primii trei și ultimii trei elevi în ordinea mediilor din școală;

Sarcini pentru realizarea unui mini-proiect

Imaginați cereri de informații specifice proiectului vostru pentru a folosi indexarea (în sens descendent, cu expresii compuse din mai mulți termeni de tipuri diferite) și căutarea rapidă.

Relaționarea tabelelor

- Tipuri de relații
- Crearea și ștergerea unei relații

Tipuri de relații

Într-o bază de date relațională, așa cum am văzut în lecțiile anterioare, se pot stabili între tabele numai relații de tipurile 1-1 și 1-n.

Relația 1-1 este stabilită între o tabelă numită **părinte** și o alta numită **copil** prin intermediul unui câmp sau al unei expresii comune. Atunci când pointerul de fișier se deplasează în tabela părinte, pointerul fișierului copil se poziționează automat pe primul articol care are valoarea expresiei de legătură egală cu cea din fișierul părinte. Aceeași tabelă poate fi legată de mai multe tabele.

Funcționarea relațiilor 1-n este următoarea: la poziționarea pe o înregistrare din fișierul părinte se leagă toate articolele fișierului copil cu aceeași valoare a cheii. Trebuie îndeplinite următoarele condiții:

- Ambele tabele trebuie deschise înaintea comenzii.
- Tabela părinte trebuie să fie ultima selectată.
- Tabela copil este indexată după aceeași expresie ca și a legăturii.
- Nu sunt permise cicluri: o tabelă nu poate fi și părinte și copil al uneia și aceleiași tabele.
- Tabela poate fi legată de mai multe tabele prin chei distincte.
- Legăturile unei tabele se numerotează în ordinea definirii lor.

Crearea și ștergerea unei relații

Fixarea unei relații de tip 1-1

```
SET RELATION TO <exp1> INTO <alias1> [,<exp2> INTO <alias2>]
[ADDITIVE]
```

Comanda anterioară stabilește o legătură 1-1 între tabela activă și cea precizată prin aliasul ei în clauza **INTO** pe baza expresiei cheie **<exp1>**. Clauza **ADDITIVE** permite păstrarea noii legături definite alături de cea anterioară.

Fixarea unei relații de tip 1-n

Comanda următoare transformă relația definită anterior într-o relație 1-n.

```
SET SKIP TO [<alias1> [,<alias2>]]
```

Ștergerea unei legături

Comanda următoare permite ștergerea unei legături:

```
SET RELATION OFF [INTO <alias>
```

Funcții

Funcție	Efect
<code>=RELATION (<nr-relatie>)</code>	Returnează expresia relației date prin <code><nr-relatie></code> .
<code>=TARGET (<nr-relatie>)</code>	Returnează numele tabelii secundare legate prin relația <code><nr-relatie></code> de fișierul deschis în zona specificată.

Studiu de caz

Fie baza de date SCOALA cu tabelele Elevi, Clase, Obiecte, Profesori, Sali, a căror structură a fost analizată și proiectată în lecțiile anterioare. Vom exemplifica modalități diferite de fixare a relațiilor între tabele.

1. Fixarea relației **ELEVI—cls—(1, 1)——>CLASE**

CLASE		ELEVI					
cls	dirig	nume	pren	cls	absn	admis	
9a	barbu	1	albuleț	geo	11a	4	.T.
10a	carp	2	albuleț	ina	11a	7	.F.
11a	doltu	3	Marinică	ion	9a	0	.F.
11b	nanu	4	popescu	teo	11b	5	.T.
		5	Stan	ana	11c	6	.T.

```
use elevi in 1                                obținem pentru fiecare elev
use class in 2 order cls                      numele dirigintelui
sele 1                                        && fișierul copil trebuie indexat
set relation to cls into B
list a->cls, a->nume, b->dirig
```

2. Fixarea relației **CLASE—cls—(1, 1)——>ELEVI**

CLASE		ELEVI					
cls	dirig	nume	pren	cls	absn	admis	
9a	barbu	1	albuleț	geo	11a	4	.T.
10a	carp	2	albuleț	ina	11a	7	.F.
11a	doltu	3	Marinică	ion	11a	0	.F.
11b	nanu	4	popescu	teo	11b	5	.T.
		5	Stan	ana	11c	6	.T.

eof

```
use elevi in 1 order cls                      obținem pentru fiecare diriginte numele
use class in 2                                primului elev din clasa sa
sele 2                                        presupunem că indexul cls din ELEVI
set relation to cls into a                    este creat deja
list a->cls, a->nume, b->dirig
```

3. Fixarea relației **CLASE—cls—(1, n)→ELEVI**

&& obținem pentru fiecare diriginte lista elevilor săi

```
use elevi in 1 order cls
use clase in 2
sele 2
set relation to cls into a
set skip to a
```

Pentru vizualizarea legăturii am deschis ferestrele Browse. La poziționarea pe un articol în fereastra Clase, automat în cealaltă fereastră sunt selectate doar liniile clasei respective.

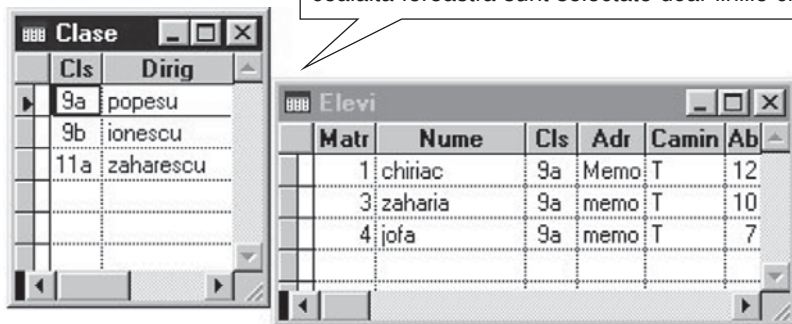


Figura 9-1: Vizualizarea legăturii

4. Fixarea relației **ELEVI—cls—(1,1)→CLASE —cls—(1, 1)→SALI**

&& Vrem să aflăm în ce sală învață elevul x

```
use elevi in 1
use clase order cls in 2
use sali order cls in 3
```

```
sele 1
set relation to cls into B
sele b
set relation to cls into C
sele 1
list c.sala for a.nume=x and a.pren=y
```

#	nume	pren	cls	cls	dirig	cls	sala
1	albuleț	geo	11a	9a	barbu	11a	1.1
2	albuleț	ina	11a	10a	carp	11b	1.2
3	marinică	ion	11a	11a	doltu	11c	2.1
4	popescu	teo	11b	11b	nanu	11d	2.2
5	stan	ana	11c				

5. Stabilirea relației **CLASE—(1,1)→SALI** prin numărul articolului

```
use clase in 1
use sali in 2
sele 1
set relation to recno() into b
list a->cls, a->diriginte,
b->sala for a.cls='12'
```

Afișăm ce diriginți au elevii claselor a XII-a și unde învață.

6. Stabilirea unei relații **ELEVI**—**n, n**—>**PROFESORI**

&& Vrem să aflăm profesorii elevului X.

Datele se găsesc în relația n-n. Un elev are mai mulți profesori, un profesor are mai mulți elevi. Codificarea unei astfel de relații se face prin intermediul unei relații noi. Vom folosi tabela **OBIECTE** care va conține numele obiectului, codul profesorului și codul clasei.

ELEVI—**cls**—(1, n)—>**OBIECTE**—**codp**—(1, 1)—>**PROFESORI**

#	nume	cls	cls	obiect	codp	codp	nume
1	Albulet Ion	11a	11a	mate	1	1	POPA
2	Albulet Jan	11a	11a	fizica	2	2	ALBU
3	Marinica B	11a	11a	chim	3	3	NICU
4	Popescu K	11b					
5	stan	11c					

```
use elevi in 1
use obiecte in 2 order cls
use profesori in 3 order codp
sele 1
set relation to cls into obiecte
set skip to b
sele 2
set relation to codp into c
sele 1
list b.obiect, c.nume for
a.nume=x
```

7. Stabilirea unei relații multiple

OBIECTE—**cls** (1-1)—>**ELEVI** ; **OBIECTE**—**codp**(1-1)—>**PROFESORI**

Vrem să afișăm pentru obiectul “Fox-Pro” elevii care studiază acest obiect și cu numele profesorului.

Varianta 1:

```
use obiecte in 1
use elevi in 2 order cls
use profesori in 3 order codp
sele 1
set filter to nume="Fox-Pro"
set relation to cls into b, codp
into c
set skip to b
list a.obiect, b.nume, c.numep
```

Varianta 2:

```
set relation to cls into b
set relation to codp into c
set skip to b
list a.obiect, b.nume, c.numep
```



sarcini de laborator

I. BIBLIOTECA

- Realizați afișarea tuturor cărților împrumutate de cititorul cu codul X.
 - Fixați relația **OPERATII**—1-n—>**CARTI**.
 - Filtrați părintele pe **cod-cititor=X**.
- Afișați numele cititorilor care au împrumutat cartea cu numărul de inventar X.
 - Fixați relația **OPERATII**—1-n—>**CITITORI**.
 - Filtrați părintele pe **nr-inventar=X**.
- Care sunt cărțile și de cine au fost împrumutate în ziua X?
 - Fixați relația **OPERATII-1-1**—>**CARTI** prin **nr-inventar**—1-1—>**CITITORI**.
 - Filtrați prin **cod-cititor** și **părintele** pe **data-împrumut =X**.

4. Ce se obține?
5. Care sunt restanțierii (numele tuturor cititorilor și ale cărților nerestituite dacă au trecut mai mult de două săptămâni de data împrumutului)?
 - *Aceeași relație ca în sarcina anterioară.*
 - *Ordonati tabela OPERATII după cod-cititor.*
 - *Filtrați tabela operații: date()-data-impr<14 and empty(data-rest).*
6. Aflați cine și când a împrumutat cărți de Mihai Eminescu.
 - *Indexați unic CARTI pe numele autorului și titlul cărții.*
 - *Fixați relația CARTI—1-n—>OPERATII—1-1—>CITITOR.*
7. Aflați care este câmpul de legătură pentru prima relație și care este câmpul de legătură pentru cea de a doua relație.
8. Aflați care este numele fișierului copil din cele două legături.

II. Informațiile despre personalul unei societăți comerciale sunt trecute în fișierele următoare.

PERS (cod-persoana, nume, prenume, adresa, data-nașterii),
 COPII (cod-persoana, prenume-copil, data-nașterii),
 SALARII (cod-persoana, salar, loc-munca, funcție) și
 RETINERI (cod-persoana, cod-reținere, suma).

Fixați relațiile și aflați:

- a) lista personalului cu numele, prenumele, funcția, salariul, suma totală a reținerilor și restul de încasat;
- b) lista alfabetică a persoanelor cu copii; pentru fiecare copil se va trece doar prenumele și vârsta;
- c) numele complet al fiecărui copil, cu data nașterii și vârsta în ani și luni;
- d) numele și adresa persoanelor cu funcția X;
- e) toate persoanele, aranjate alfabetic și sumele reținute (eventual pe tipuri de rețineri);
- f) pentru fiecare loc de muncă, numele persoanelor ce lucrează acolo, cu numărul de copii și reținerile lor.

*Atenție! Creați tabelele și încărcați-le **cu date de test**. Citiți cu atenție solicitările de informații și imaginați situațiile posibile.*

Plan

Situația legată de rețineri și copii	Fără copii	Cu un copil	Cu mai mulți copii
Fără reținere	Popescu	Ionescu	Vasilescu
Cu o singură reținere Ex. CAR	Zaharescu	Minulescu	Albulescu
Cu mai multe rețineri. (Ex. CAR, RATE, PENSIE)	Dobrescu	Enachescu	Nasuescu
Situația legată de funcții și locuri de muncă	Muncitor	Contabil	inginer
Sectia 1	Popescu	Ionescu	Vasilescu
Sectia 2	Zaharescu	Minulescu	Albulescu
Sectia 3	Dobrescu	Enachescu	Nasuescu

Sarcini pentru realizarea unui mini-proiect

Pentru fiecare bază de date proiectați relațiile dintre tabele și imaginați interogări pe care să le rezolvați folosind aceste relații.

Transferul de date între tabele Visual FoxPro și alte structuri

- Importarea și exportarea datelor din și către masive de memorie
- Importarea și exportarea datelor din și către fișiere de alte tipuri

Importarea și exportarea datelor din și către tablouri de memorie

De multe ori, folosirea tablourilor de date este preferabilă folosirii fișierelor de date, deoarece viteza de accesare a memoriei este net superioară vitezei de accesare a discului. De exemplu, sortarea unui masiv este mult mai rapidă decât sortarea unui fișier. Comunicarea între bazele de date și tablouri se realizează în ambele sensuri prin comenzile corespunzătoare. Tabelul următor prezintă comenzi și funcții utile:

Comanda	Efect
APPEND FROM ARRAY <tablou> [FOR<cond>] [FIELDS <lista-cmp>]	Se adaugă la tabela activă liniile dintr-un tablou; fiecare linie corespunde unei înregistrări; coloanele sunt copiate în ordinea câmpurilor; se ignoră elementele în plus ale matricii; câmpurile în plus sunt completate automat cu valori vide. Se face conversia la tipul câmpului.
GATHER FROM <tablou>/MEMVAR [FIELDS <lista-câmp>] [MEMO]	Are loc copierea vectorului <tablou> sau a unor variabile de memorie cu același identificator (MEMVAR) în înregistrarea curentă din tabela activă. Copierea se face de la stânga la dreapta în ordinea coloanelor, dacă nu este precizată clauza FIELDS . Clauza MEMO este necesară dacă printre câmpurile de copiat este și un câmp Memo.
COPY TO ARRAY <tablou> [FIELDS <lista câmpuri>] [<domeniu>] [FOR<cond>] [WHILE<cond>]	Comanda realizează trecerea articolelor din tabela activă într-un tablou, astfel încât fiecare articol devine o linie. Implicit sunt considerate toate câmpurile. Matricea nu trebuie să fie declarată anticipat.
SCATTER [FIELDS <lista-câmp>] [memo] TO <tablou>/MEMVAR	Câmpurile (toate sau cele specificate în FIELDS) din tabela activă se vor copia într-un vector sau în variabilele speciale (clauza MEMVAR).
REPLACE FROM ARRAY <tablou> [FIELDS<lista câmpuri>] [<domeniu>] [FOR <cond>] [WHILE <cond>]	Se înlocuiesc cu elementele din tabloul specificat acele valori care corespund câmpurilor din clauza FIELDS (implicit toate câmpurile). Comanda REPLACE are ca domeniu implicit articolul curent.

Exemple

1. Adăugați la tabela ELEVI elementele masivului NOTE.

ELEVI.DBF					note [2, 6]					
nume	rom	mat	fiz	chim	x	8	8	7	8	10
ION	10	8	8	6	y	10	9	8	9	10
x	8	8	7	8	← append from array note					
y	10	9	8	9						

2. Copiați în matricea Note toți elevii cu nota 10 la română. Matricea NOTE nu este necesar să fie declarată anticipat. Ea va avea ca dimensiuni numărul de linii egal cu numărul de articole filtrate, iar numărul de coloane egal cu numărul de câmpuri.

ELEVI.DBF					note [2, 6]					
nume	rom	mat	fiz	chim	ION	10	8	8	6	.F.
ION	10	8	8	6	POPA	10	9	8	9	.F.
LUCA	8	8	7	8	← copy to array note for rom=10					
POPA	10	9	8	9						

3.

```
Use elevi
scatter to array a
? nume, a[1], a[2]
ION ION 10
```

Se introduc date despre primul elev în vectorul A.
Numărul de elemente este egal cu numărul de câmpuri.
Putem să ne referim la elemente prin indicii lor.
Observați folosirea câmpului „nume“!

```
go bottom
```

Are loc poziționarea la ultimul articol.

```
scatter memvar
```

Este creat un șir de variabile cu aceleași nume ca ale câmpurilor cu valori din ultimul articol.

```
?m.nume, m.mat, m.rom
POPA 9 10
```

Observați calificarea variabilelor!

4. La examenul de bacalaureat elevii de la „Informatică“ au fost împărțiți în trei comisii, fiecare comisie organizându-și singură examenul și evidența concurenților. Se cere concatenarea datelor în vederea calculării tuturor mediilor și afișării tuturor rezultatelor.

nume	p1	p2	proiect
albu	8.77	7.63	10
andrei	10	10	10
aurel	5.66	6.50	9

COMISIE1.DBF

p1 reține nota la română,
iar p2 – nota la matematică.

nume	mate	lrom	med
barba	10	9	9.50
baciu	10	7	8.50
bucur	10	7	8.50

COMISIE2.DBF

Numele câmpurilor corespund
obiectelor de examen

candidat	proba1	proba2	școala
carp	10	4.66	economic
cărbune	10	8.99	informatică
ciurea	10	5	informatică
cocea	10	10	economic

COMISIE3.DBF
 Proba1 = română,
 Proba2 = matematică

Având nume de câmp diferit, nu vom putea folosi comanda **Append from**. O variantă de rezolvare este copierea datelor în tablouri în ordinea dorită a câmpurilor și apoi adăugarea într-o nouă tabelă BAC.

```

use comisie1 in 1
use comisie2 in 2
use comisie3 in 3
select 4
create dbf bac (nume C(10), ;
rom N(5,2),mate N(5,2), ;
med N(5,2))
sele 1
copy to array a
sele 2
copy to array b fields nume, ;
lrom, mate

sele c
copy to array c fields
candidat, ;
proba2, proba1;
for scoala='informatic'
sele d
append from array a
append from array b
append from array c
close databases
release all

```

Importarea și exportarea datelor din și către alte tipuri de fișiere

Transferul informațiilor între produse program sau, mai bine spus, între fișiere baze de date tip Xbase și alte tipuri de fișiere se poate face în ambele sensuri prin comenzi speciale.

Comanda	Efect
APPEND FROM < fisier> TYPE <tip-fis>	Trecerea datelor din fișierul de tip special în tabela activă.
COPY TO <fisier> TYPE <tip-fis>	Transferul datelor dintr-o tabelă .DBF în alt tip de fișier.
IMPORT FROM <fisier> TYPE PDOX / RPD / WKS / WRK / XLS	Conversia unui fișier de alt tip într-un fișier .DBF.
EXPORT TO <fisier> FIELDS <lista-câmpuri> [<domeniu>] [FOR <cond>] [WHILE <cond>] [TYPE] DIF / MOD / WKS / WRK / XLS	Conversia unui fișier .DBF în alt tip de fișier. Tipurile de fișiere sunt indicate în clauza type . Datele care vor fi convertite pot fi filtrate /selectate.

<Tip-fis> poate fi:

- **DELIMITED / DELIMITED WITH <car>** atunci când fișierul sursă este în format ASCII sau are câmpurile delimitate de <car>.
- **DELIMITED WITH BLANK** pentru fișiere cu câmpuri delimitate de spații.
- **SDF** pentru fișier ASCII format data sistem (**EXTENSIE .TXT**).
- **WKS** pentru fișier LOTUS 1-2-3 etc.

Exemple

Fie tabela ELEVI, cu următorul conținut:

#	nume	dn	cls	absn	absn	m1	m2	m3	an	bi
1	popescu		11a	10	15	6	7	8	8	
2	ionescu		11a	0	5	9	7	8	8	
3	zaharescu		11b	12	1	2	10	10	10	
4	fictiv		11b							
5	albu		11a	10	10	10	10	10	10	

Dorim o operație de copiere a tabelii elevi într-un fișier `.TXT` având ca separatori de câmpuri virgula, iar șirurile delimitate prin ghilimele.

```
copy to xxx delimited
type xxx. txt
```

```
"popescu", ",", "11A", 10, 15, 6, 7, 8, 8, ""
"ionescu", ",", "11A", 0, 5, 9, 7, 7, 8, ""
"zaharescu", ",", "11B", 12, 1, 2, 10, 10, 10, ""
"fictiv", ",", "11B", , , , , , , , , ""
"albu", ",", "11A", 10, 10, 10, 10, 10, 10, , ""
```

Observați clauza
Delimited!

<code>append from xxx delimited</code>	Adăugarea datelor din <code>xxx.txt</code>
<code>go 6</code>	Poziționare pe articolul 6
<code>dele rest</code>	Ștergerea celorlalte articole.
<code>pack</code>	
<code>copy to zzz type sdf</code>	Exportarea bazei sub formă standard.
<code>type zzz.txt</code>	

```
popescu 11A1015 6 7 8 8
ionescu 11A 0 5 9 7 8 8
zaharescu 11B12 1 2101010
fictiv 11B.. ..
albu 11A101010101010
```



sarcini de laborator

Ce execută secvențele următoare?

Secvența 1
Use elevi
scatter to memvar
accept to nume
input to rom
input to mat
fiz=elevi.fiz
append blank
gather from memvar

Secvența 2
Use elevi
scatter memvar
accept to nume
go bottom
fiz=elevi.mat+1
append blank
gathr memvar

Secvența 3
Use elevi
go 2
scatter memvar
?memvar (1)

Secvența 4
Use elevi
goto 2
scatter to
memvar
?m.nume, m.mat

Prelucrări statistice și financiare

- Numărarea articolelor: **COUNT**
- Însumarea valorilor unor câmpuri: **SUM**
- Calculul mediei aritmetice: **AVERAGE**
- Diverse calcule statistice și financiare: **CALCULATE**
- Totalizarea valorilor: **TOTAL**
- Funcții financiare

Scopul principal pentru care sunt create bazele de date este obținerea într-un timp cât mai scurt a unor informații cu privire la datele conținute în tabele. Aceste informații pot fi de natură diferită, mai detaliate sau mai sintetizate, sub formă de liste, tabele sau simple valori, informații statistice, totalizatoare.

Numărarea articolelor: **COUNT**

```
COUNT [TO <var>] [<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Comanda numără articolele din tabela activă și, dacă este prezentă clauza **TO <var>**, depune rezultatul în variabila specificată.

Exemplu. Fie tabela ELEV1.DBF cu următorul conținut:

nume	cls	absm	absn	p1	p2
popescu minel	11B	10	5	10	10
albuleț geo	11A	4	4	10	10
popescu sile	11C	5	6	10	10
marinică marin	11C	10	6	10	9
ionesco marin	11C	4	6	10	9
albuleț dino	11A	0	0	10	9

Numărul total de elevi se calculează în variabila V
Count to V
?V

Însumarea valorilor unor câmpuri: **SUM**

```
SUM [<lista -exp>] [TO <lista-var>/TO ARRAY <tablou>]  
[<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Comanda permite însumarea valorilor existente în articolele selectate, conform expresiilor precizate.

Exemplu: Considerăm același fișier din exemplul anterior. Urmăriți efectul următoarelor comenzi!

sum	&& în lipsa clauzelor se însumează valorile
absn absm p1 p2	&& tuturor câmpurilor numerice și rezultatele
27 33 60 57	&& se afișează pe ecran
Sum for cls='11A' absm +absn	&& se afișează pe ecran doar
8	&& valoarea calculată

Sum for cls='11A'
absm +absn to x, v

&& eroare: număr de variabile
&& mai mare

Calculul mediei aritmetice: **AVERAGE**

```
AVERAGE [<lista-exp>] [TO <lista-var>/TO ARRAY <tablou>]  
[<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Comanda permite calculul mediei aritmetice a valorilor expresiilor din <lista-exp>, pentru articolele din tabela activă care se încadrează în domeniul precizat și îndeplinesc condițiile din FOR și WHILE. Parametrul TO <lista-var> conține lista de variabile în care vor fi depuse valorile mediilor calculate.

Exemplu

```
Average                && în lipsa clauzelor se calculează media valorilor  
4.50 5.50 10.9        && tuturor câmpurilor numerice  
-----  
go top                 && calculăm media absențelor  
average absm while cls='11B' && primilor elevi din clasă
```

Diverse calcule statistice și financiare: **CALCULATE**

```
CALCULATE [<lista-exp>] [TO <lista-var> /TO ARRAY <tablou>]  
[<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Comanda poate calcula valorile mai multor expresii și depune rezultatul într-o listă de variabile sau într-un tablou.

În alcătuirea unei expresii pot intra următoarele funcții:

1. **AVG(<expn>)**: permite calculul mediei aritmetice a expresiei <expn> ce poate conține câmpuri numerice ale tabelii active.
2. **CNT()**: permite numărarea articolelor selectate din tabela activă.
3. **SUM(<expn>)**: permite însumarea valorilor expresiei <expn>, expresie ce conține câmpuri numerice ale tabelii.
4. **MAX(<exp>)**: extrage cea mai mare valoare a expresiei <exp> calculată pentru fiecare articol selectat al tabelii active.
5. **MIN(<exp>)**: extrage cea mai mică valoare a expresiei calculată pentru fiecare articol selectat al tabelii active.
6. **STD(<exp>)**: returnează abaterea medie pătratică a expresiei date. Cu cât abaterea medie pătratică este mai mică, cu atât valorile sunt mai apropiate de medie.
7. **VAR(<exp>)**: calculează dispersia expresiei (pătratul abaterii medii pătratice).

Exemplu: Fie tabela ELEVI.

Calculăm numărul de articole, suma absențelor, media și numărul maxim al absențelor nemotivate:

```
calculate cnt(), sum(absn), avg(absn), max(absn)  
6 27 4. 50 6
```


Totalizarea valorilor: TOTAL

```
TOTAL ON <cheie> TO <fis.dbf> [FIELDS <lista-câmp>]
[<domeniu>] [FOR <cond>] [WHILE <cond>]
```

Comanda **TOTAL** realizează o nouă tabelă, numită **<fis.dbf>**, cu aceeași structură ca a tabelii active. Tabela activă este parcursă în întregime și pentru fiecare grup de articole care au aceeași valoare a expresiei **<cheie>** se adaugă câte un articol în tabela **<fis.dbf>**. Articolul are cheia unică a grupului și suma valorilor din câmpurile specificate în clauza **FIELDS** (sau toate câmpurile numerice dacă lipsește această clauză).

Exemple

1. Fie tabela ELEVII din exemplul anterior. Dorim obținerea totalului de absențe (motivate și nemotivate) pe clase.

```
use elev
index on cls tag cls
total on cls to man fields absm, absn
use man
list cls, absm, absn
```

2. În fișierul ELEVII avem trecute notele la diferite probe de examen. Dorim să mai adăugăm două articole: un articol cu mediile și altul cu notele maxime la fiecare probă.

Nume	Proba1	Proba2	Proba3
Popescu	10.00	3.78	8.00
...
Medii			8.75
Nota maxima			10

mediile la aceste probe

nota maximă la aceste probe

```
use elevi
declare a[3], b[3]
average all proba1, proba2, proba3 to array a
calculate max(proba1), max(proba2), max(proba3) to array b

append blank
gather from a fields proba1,;
    proba2, proba3
repl nume with 'medii'

append blank
gather from b fields proba1,;
    proba2, proba3
repl nume with 'nota maxima'
```

Funcții financiare

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

Funcția calculează valoarea viitoare a unei depuneri regulate **<expN1>** cu o creștere constantă în cadrul unei investiții, cu o dobândă fixă **<expN2>** pe o perioadă dată **<expN3>**.

Exemplu: Presupunem că am deschis un cont la o bancă ce practică o dobândă de 1% lunar. Suma lunară pe care dorim să o depunem este de 3000 unități monetare. Ne interesează ce sumă vom avea în cont după 36 luni:

```
? fv(3000, 0.01, 36) && 129230.64
PV(<expN1>, <expN2>, <expN3>)
```

Funcția calculează valoarea la zi a unei investiții constituite printr-un vărsământ regulat cu o sumă constantă $\langle \text{expN1} \rangle$, de-a lungul unui număr de perioade date $\langle \text{expN3} \rangle$, la o dobândă fixată $\langle \text{expN2} \rangle$.

Exemplu: Presupunem că avem un cont la o bancă ce practică o rată a dobânzii de 1% lunar. Ne interesează ce sumă am putea acoperi din acest cont în 4 ani, plăind lunar câte 4000 lei. Deci 4000 este suma ce se scoate regulat (lunar) din cont; 0,01 este dobânda băncii, 48 este numărul de luni.

? `pv(4000, 0.01, 48) && 151895.84`

PAYMENT ($\langle \text{expN1} \rangle$, $\langle \text{expN2} \rangle$, $\langle \text{expN3} \rangle$)

Funcția calculează mărimea rambursărilor constante efectuate la intervale regulate care permit amortizarea unei sume $\langle \text{expN1} \rangle$, cu dobândă constantă $\langle \text{expN2} \rangle$, pe un număr dat de perioade $\langle \text{expN3} \rangle$.

Exemplu: Presupunem că avem de returnat un împrumut de 500.000 unități monetare. Ce sumă vom plăti lunar pentru ca în 2 ani să achităm împrumutul? Știm că depunerile lunare se adaugă într-un cont la o bancă ce acordă o dobândă lunară de 1%.

? `payment(500000, 0.01, 24) && 23536.74`



sarcini de laborator

I. O societate comercială înregistrează repartiția produselor date spre vânzare la diferite magazine în tabela REPART cu structura (data, cod-produs, nume, preț-livrare, grupa, magazin, cant, um). Magazinele sunt codificate numeric de la 1 la 5. Grupele de produse sunt codificate pe două caractere, de exemplu: „TE”=textile, „AL”=alimente, „EL”=electrice ș.a.

data	cod	nume	pret	grupa	mag	cant
01.01.95		tele color	1500	el	1	10
01.01.95		combina	2300	el	2	2
01.02.95		casetofon	1600	el	1	15

- Codificați produsele astfel încât să conțină grupa, primele 5 caractere din numele produsului și prețul.
&& construim o expresie din câmpurile grupă (character), nume-produs (character) și prețul (numeric)! Ce funcție trebuie folosită, val sau str???
- Aflați pentru fiecare grupă de produse numărul și valoarea produselor distincte distribuite la toate magazinele în data X.
&& facem un artificiu și înlocuim peste tot câmpul cod cu valoarea 1
&& indexăm după grupa unic
&& totalizăm pe grupe
&& în câmpul cod – fictiv umplut cu valoarea 1 – vom obține numărul dorit.
- Afișați câte produse distincte comercializează unitatea.
&& Indexare unică și numărare

- Calculați pentru fiecare cod-produs cantitatea totală repartizată în întreaga perioadă de evidență.
 && vom indexa după cod-produs
 && vom folosi comanda TOTAL pentru câmpul cant
- Calculați pe fiecare magazin din subordine valoarea produselor repartizate.
- Aflați cel mai scump produs.
 && Index on str(-pret)+nume to x unique
 && X= pret
 && List nume while pret=x
- Treceți într-un istoric (fișier de tip text) repartițiile din perioada anterioară datei X, separând coloanele prin virgulă, delimitator de șir fiind ghilimelele.

II. Vânzări. Evidența vânzărilor la mai multe magazine ale aceleiași societăți comerciale „SC INTIM SRL“ este ținută în baza de date VANZARI (dată, cod-magazin, cod-raion, cod-produs, um, cantitate). Aflați:

- Câte raioane au vândut azi produsul x?
- Care este cantitatea totală din produsul x vândută azi?
- La câte magazine se găsesc raioane de papetărie (cod=2)?
- Care este numărul total de magazine ale societății și numărul de raioane distincte?
- Care este totalul cantității vândute din produsul x în fiecare magazin?
- Care este valoarea totală a vânzărilor pe lista de raioane (fiecare raion o singură dată, chiar dacă el aparține unor magazine diferite)?
- Care este valoarea medie a vânzărilor pe magazine?
- Care este produsul vândut cel mai scump și unde s-a vândut el?
- Care este primul moment când s-a vândut produsul x?
- Care sunt cele 3 magazinele fruntașe (cu cele mai mari vânzări în ultima lună)?

III. Fișierul PERSONAL are informații despre salariații unei societăți: (cod, nume, loc-muncă, grad, funcție, salariu, impozit, rețineri, sporuri).

- Aflați:
 - salariul mediu pe locuri de muncă;
 - salariul mediu pe funcții;
 - fondul total de salarii;
 - numărul de salariați pe locuri de muncă;
 - impozitul total, reținerile totale, totalul sporurilor pe locuri de muncă;
 - funcția și numele persoanei cu salariul maxim.
- Să presupunem că avem grila de impozitare a salariaților în tabloul IMPOZ reținut în fișierul IMPOZ.MEM pe directorul curent. Modificați această grilă conform noilor reglementări (precizați reglementările!).

Grad /salariu	4-10	10-20	>20
gr1	10%	25%	30%
gr2
def	5%	15%	20%

Numerele reprezintă sute de RON.

Corecționați tabela PERSONAL cu impozitul calculat conform noii grile! Procentul de impozitare se aplică la salariu!

RECAPITULARE

Operații elementare asupra tabelor în Visual FoxPro

Pentru lucrul cu structura:	CREATE TABLE / CREATE MODIFY / COPY STRUCTURE SET FIELDS TO, =FIELDS(N)
Deschidere/închidere	USE, CLOSE ALL SELECT <n>
Afișarea conținutului	LIST, DISPLAY - afisare continut SET FILTER TO, =RECNO(), =RECCOUNT()
Căutare și poziționare	LOCATE, CONTINUE, FOUND() FIND / SEEK
Ordonarea conținutului	SORT, INDEX
Duplicarea conținutului unei tabele	COPY TO <dbf> / TO TYPE / TO ARRAY SCATTER
Adăugarea datelor	APPEND / APPEND BLANK APPEND FROM ARRAY / <dbf> / TYPE GATHER
Ștergerea datelor	DELETE, RECALL, PACK, ZAP, SET DELETED, =DELETED()
Corecția datelor	REPLACE, BROWSE
Relaționarea tabelor	SET RELATION TO - fixare relatie 1-1 SET SKIP TO - fixare relatie 1-n

Care dintre următoarele secvențe creează o copie (fidelă!) a fișierului BANCHERI (cod, nume, adr, afaceri) în fișierul tabelă cu numele BAN?

1. copy file bancheri.dbf to; ban.dbf copy file bancheri.fpt to ban.fpt	2. use bancheri sort on nume; to ban; field nume	3. use bancheri copy to array; ban	4. use bancheri copy stru to; ban
5. use bancheri copy stru to ban copy to array to ban use ban append from array ban release ban	6. use bancheri copy to ban; for not empty(cod)	7. use bancheri sort on nume; to ban	8. use bancheri index on nume; tag ban
9. use bancheri list to file; ban	10. use bancheri copy stru to ban use ban append from; bancheri	11. use bancheri copy to ban	12. use bancheri copy file; bancheri.dbf to; ban.dbf

Fie un fișier de date numit CONTRACTE, creat la 11 decembrie 2000, cu toate contractele încheiate de o societate comercială cu diverși furnizori pentru livrarea produselor necesare activității sale.

nrc	data	furnizor	produs	cant	pret	term-livr	onorat	conditii
n, 5	d, 8	c, 20	c, 10	n, 10, 2	n, 10	D, 8	l, 1	m, 10

unde: **nrc**=numărul contractului, **data**=data semnării contractului; **term-livr**=termenul de livrare; **onorat**=arată dacă a fost sau nu onorat contractul, **condiții**=alte condiții de livrare.

Cerințe:

- Listati contractele încheiate cu „SC INTIM SRL“ în anul curent.
- Aflati ce produse și în ce cantități vor veni azi și de la cine.
- Afișati valoarea fiecărui contract (presupunând că pe un contract este un singur produs!) al furnizorul X.
- Afișati contractele cu termen de livrare depășit. Puteți număra câte contracte au termenul depășit fără să folosiți comenzile de calcul?
- Afișati ultimele 5 articole existente în tabelă.
- Aflati câte contracte are furnizorul X. (Nu folosiți comenzi de calcul!)
- Aflati dacă există vreun contract cu furnizorul X. Dacă da, afișati numărul și data acestui contract. Cum am putea afla și următorul contract al aceluiași furnizor?
- Separati în două fișiere conținutul tabelului CONTRACTE. Astfel, tabela **FURNIZ .DBF** trebuie să rețină datele din câmpurile Nrc, Data, Furnizor și Onorat și **PRODUSE .DBF** va reține datele din câmpurile Nrc, Data, Produs, Cant, Pret, Term_livr.
- Deschideți în două zone tabelele FURNIZ și PRODUSE. Afișati toate produsele contractate de furnizorul X în primul său contract.
- Afișati *la imprimantă* următoarea situație, scriind titlul și capul de tabel pe prima foaie a raportului.

LISTA CONTRACTELOR LA DATA DE >>>>

nr-contract	furnizor	valoare	termen

Punctaj: Se acordă câte 0,9 puncte/subiect; se acordă 1 punct din oficiu.

Testul 2**Lucrul cu mai multe fișiere, actualizarea datelor**

Considerăm un sistem de evidență ținut pe calculator al ocupării camerelor la mai multe unități de cazare din diferite zone turistice, cu următoarea bază de date:

Fișierul UNITATI are înregistrate toate unitățile de cazare:

UNITATI.DBF

cod	nume	fel	categorie	agentie
n, 5	c, 15	c, 1	n, 2	c, 15

numele unității de cazare și stațiunea:
ex.: „mamaia/alfa“,
„eforie/gama“

H=hotel
M=motel
V=vilă
C=căsuțe

Numele agenției
care a închiriat
unitatea

Fișierul CAMERE reține toate camerele din toate unitățile de cazare, pentru fiecare indicându-se numărul de paturi și starea de ocupare.

cod_unit	cod_cam	nr_pat	are_tel	are_tv	Pret	este_ocup
n, 4	n, 3	n, 1	l, 1	l, 1	n, 5	l, 1

Fișierul OCUPARE ține evidența sosirii și a plecării tuturor turiștilor în anul curent, codul camerei și al hotelului.

cod_unit	cod_cam	data_s	data_p	nume_pers	b_i
n, 4	n, 3	d, 8	d, 8	c, 15	c, 10

Scrieți succesiunea de comenzi necesare pentru:

- 1) Înregistrarea unui turist nou.
- 2) Plecarea unui turist.
- 3) Mărirea cu 10% a prețului camerelor cu telefon.
- 4) Ștergerea logică a camerelor hotelului X care intră în „deratizare“.
- 5) Modificarea apartenenței unei unități de cazare la o agenție de turism.
- 6) Preluarea telefonului din camera C1 în camera C2 a hotelului X.
- 7) Este dezafectată camera 5 din hotelul „DELTA“ stațiunea „MAMAIA“. Persoanele care locuiesc în această cameră se vor muta în camera 19.
- 8) Persoana X se mută într-o altă cameră liberă, cu două paturi, televizor și telefon din hotelul „DELTA“.
- 9) Trecerea într-o evidență separată a tuturor camerelor cu patru paturi, care nu sunt ocupate la data curentă.

Punctaj: Se acordă 1 punct/subiect. Se acordă 1 punct din oficiu.

Testul 3 Definierea structurii unei baze de date pornind de la cerințe

Citiți cu atenție următoarele cerințe și proiectați structura bazei de date necesare:

1. Afișați toate piesele de teatru programate pentru orașul X, în perioada d1-d2 sub numele „afiș teatral pentru localitatea...”.
2. Unde (în ce localități) a avut spectacole trupa de teatru X, când?
3. Toate spectacolele programate în sala „Majestic“ după 1 decembrie vor începe la ora 18 în loc de 19, și la ora 11 în loc de ora 10.
4. Se anulează toate spectacolele cu piesa X, programate după data Y.
5. Sala X intră în reparație capitală la 1 decembrie. Reprogramați spectacolele în sala Y cu o săptămână mai târziu.
6. Ce piese de teatru are trupa „MASCA“ în stagiunea aceasta?
7. Unde (la ce teatre) s-a montat piesa X, când, în regia cui?
8. Câte reprezentații a avut piesa X, în toate montările ei?
9. Ce actori are angajați teatrul Y?
10. Ce distribuție are piesa X, de la teatrul Y, în data Z?
11. Mai sunt locuri la piesa X?
12. Care a fost cea mai de succes montare a piesei X, la ce teatru, de care trupă, în ce regie, în ce stagiune, cine a fost în distribuție?
13. Care este indicatorul de ocupare a sălii X (total locuri ocupate/total locuri disponibile) la spectacolul din data Y?
14. Câte bilete au fost vândute la un spectacol X?
15. S-a mai jucat piesa X, la alte teatre, stagiuni? Când? Unde? În ce regie?

Punctaj: Se acordă câte 0,6 puncte/subiect și un punct din oficiu.

Testul 4 Comenzile și funcțiile uzuale

Fie următoarea listă a fișierelor existente pe directorul curent:

elevi.dbf, elevi.cdx, fisier.prg, elevi.txt

- A. Pentru comenzile următoare indicați fișierele, extensia fiecăruia și extensia implicită, eventual semnalăți erorile:

<code>use elevi</code>	<code>Copy to alfa for nume=a</code>
<code>Sort on nume+str(med) to beta</code>	<code>Type beta</code>
<code>Index on nume tag delta unique</code>	<code>Use delta</code>
<code>List next 5 to file delta</code>	<code>Store "popa" to nume</code>
<code>Save to nume</code>	<code>Append memo to fisier</code>
<code>Append from fisier</code>	<code>Count to fisier</code>
<code>Sum to fis</code>	<code>Do fisier</code>
<code>Rename elev to student</code>	<code>Set default to scoala</code>
<code>Restore from date</code>	<code>Replace elevi to studenti</code>

B. Care sunt comenzile corespunzătoare următoarelor operații?

Editarea unui fișier text	Afișarea unui text
Ștergerea de pe disc a unui fișier	Citirea unei variabile
Modificarea lungimii unui câmp	Suma valorilor unui câmp din toată tabela
Afișarea unei expresii	Afișarea la imprimantă a codului sursă a unui program
Crearea unui index	Duplicarea conținutului unei tabele
Schimbarea directorului curent	Apelul unui subprogram
Adăugarea unui articol vid	Ștergerea tuturor variabilelor
Pauză în program până la apăsarea unei taste	Fixarea unei relații de tip 1-n

C. Care dintre următoarele comenzi au în formatele lor clauzele de selecție și filtrare?

REPLACE	COPY	LIST	APPEND	SUM
CALCULATE	CLOSE	DELETE	FIND	SAVE
PACK	USE	SORT	COUNT	APPEND
ERASE	RESTORE	TOTAL	ACCEPT	WAIT

D. Scrieți numele funcției care realizează acțiunea:

1. conversia de la șir la număr
2. conversia de la număr la șir
3. conversia de la dată calendaristică la șir
4. conversia de la șir la dată calendaristică
5. minimul dintre două expresii
6. rezultatul operației de căutare
7. testul asupra sfârșitului de tabel
8. numărul de articole ale tabelii active

Punctaj: Se acordă câte 2 puncte pentru fiecare subiect; se acordă două puncte din oficiu.

Testul 5

Considerăm o agenție de rezervare a biletelor de avion. Creați o aplicație care permite:

- a. Afișarea unui program al curselor interne regulate și excepționale, cronologic după ora plecării. Pentru cele cu plecări excepționale sau neregulate, se va afișa data plecării numai pentru o perioadă de 3 luni față de momentul afișării.
- b. Aflarea curselor interne pentru localitatea X, pe companii de zbor și cronologic. Se vor afișa pentru fiecare cursă: numărul, numele companiei de zbor căreia îi aparține avionul și, dacă este cursă regulată, se va trece ziua și ora, iar dacă este o cursă neregulată se va obține data și ora plecării.
- c. Afișarea unui program cronologic al curselor internaționale; plecările și sosirile de pe și pe aeroportul deservit de agenția respectivă vor fi afișate separat.

- d. Aflarea datei și orei de plecare, a datei și orei de sosire la destinație pentru o cursă dată prin numărul ei. Dacă este o cursă regulată, se va afișa un text „zilnic/săptămânal/lunar/anual“ în funcție de ritmicitatea zborului, apoi ora/ziua din săptămână și ora/data din lună, apoi ora/luna, data din lună și ora.
- e. Aflarea numărului de locuri ocupate/rezervate/libere pentru o anumită cursă, dată prin numărul ei și data zborului.

Sarcini: Proiectați baza de date și scrieți comenzile necesare obținerii informațiilor solicitate.

Punctaj: Se acordă câte 2 puncte la sarcinile a-d, 1 punct la sarcina e și un punct din oficiu.

Testul 6

Pentru activitatea de distribuire a unor mărfuri, o companie încheie contracte cu diferiți clienți, iar în contul acestor angajamente livrează marfa pe bază de facturi care trebuie achitate. Desigur, o factură se referă la un singur contract, dar pentru același contract pot fi înregistrate mai multe facturi. O chitanță – documentul care certifică achitarea unei sume – se referă la o anumită factură, dar pentru o factură pot fi înregistrate mai multe chitanțe.

Fie o bază de date cu tabelele următoare:

- a. CLIENTI, cu numele, adresa, codul, alte-inf despre clienții unei firme;
- b. CONTRACTE, cu numărul și data semnării, codul clientului, valoarea contractului;
- c. FACTURI, cu numărul și data facturării, numărul contractului pentru care s-a facturat și valoarea facturii;
- d. CHITANTE, cu numărul și data chitanței, numărul facturii, suma achitată.

Se cere:

- 1. Definiți tabelele și relațiile dintre ele. Atenție, fixați corect cheile primare și cele străine.
- 2. Aflați dacă există vreun client care nu a achitat nici o factură.
- 3. Care sunt facturile clientului X. Pentru fiecare factură se va afișa un text anunțând dacă are sau nu achitări.
- 4. Pentru fiecare chitanță înregistrată azi, afișați numele clientului și numărul contractului, numărul facturii și suma.

Punctaj. Se acordă câte 2 puncte/subiect. Se acordă 2 puncte din oficiu.

Toți senatorii și deputații trebuie să declare venitul și proprietățile pe familie. Au fost create următoarele fișiere: PARTIDE (cod_pers, nume_persoană, data_nașterii, sex, ocupație_anterioară, studii, nume_partid, funcția_in_partid, data_inscrierii, alte_date); PARLAMENT (cod_pers, tip, venit_anual_declarat), cu informații despre parlamentari (unde tip="s"=senator, tip="d"=deputat); IMOBILE (cod_pers, imobil-adresa, număr_camere, valoare).

1. Faceți următoarele corecții:

Popescu a trecut la PRM. Ionescu, fiind prieten cu Zaharescu, se înscrie la partidul acestuia din urmă. X a decedat și va fi scos din evidență, dar, înainte i se trec datele personale într-un fișier istoric cu aceeași structură ca VIP.DBF. În locul lui X intră în parlament pe aceeași funcție persoana Y din același partid.

2. Afișați sub forma următoare:

Lista parlamentarilor la data de					
Deputați			Senatori		
nume	partid	funcție	nume	partid	funcție

3. Dublați venitul senatorilor numai dacă de la ultima majorare a trecut cel puțin o lună. Ultima modificare se găsește în fișierul DATE_IMP.MEM sub numele DUA (Data Ultimei Actualizări).

4. S-a anunțat că venitul trebuie exprimat în dolari; știind cursul de schimb, faceți corecțiile necesare.

5. Aflați:

- Pentru fiecare imobil, din ce partid face parte proprietarul?
- Care sunt membrii senatori sau deputați din partidul X?
- Există vreun membru al partidului X fără casă?
- Care sunt cei mai bogați parlamentari (primii 10)?

Punctaj. Se acordă câte 2 puncte pentru subiectele 1, 2, 3, 5, un punct pentru subiectul 4 și un punct din oficiu.



sarcini pentru realizarea unui mini-proiect

Lucrați la proiectele pe echipe. Imaginați sarcini pentru folosirea indecșilor și a relațiilor dintre tabele. Codificați operații de calculare a sumelor, mediilor, diverse totalizări. Folosiți importuri de date din Excel. Realizați exportarea unei table Visual FoxPro într-o foaie de calcul Excel.

Programarea clasică în FoxPro

- Comenzi pentru structuri de control
- Proceduri și funcții utilizator
- Depanarea programelor

Dezvoltarea programelor în maniera clasică presupune folosirea tehnicilor de structurare și modularizare a programului. FoxPro permite descrierea structurilor de control și a procedurilor și funcțiilor utilizator.

Structura alternativă și comanda **IF . . .ENDIF**

```
IF <cond>
  <secv.1>
  [ELSE
  <secv.2>]
ENDIF
```

Operația de ramificare a algoritmului în funcție de valoarea de adevăr a unei expresii logice, cu revenirea, în ambele situații, într-un singur punct, se realizează cu structura alternativă codificată prin comanda **IF**.

Structura selectivă și comanda **DO CASE . . .ENDCASE**

```
DO CASE
  CASE <conditie1>
    <secv1>
  CASE <conditie2>
    <secv2 >
  CASE <conditieN>
    <secvn>
  [OTHERWISE
    <secvm>]
ENDCASE
```

Este evaluată prima condiție <c1> și, dacă este adevărată, este executată secvența de comenzi <secv1> și se părăsește structura **CASE**. Dacă nu este îndeplinită condiția <c1>, se va testa condiția notată <c2> ș.a.m.d. Se ajunge, deci, la testarea condiției <cn> atunci când niciuna dintre condițiile precedente nu a avut valoarea adevărat. Codificarea structurii selective se face prin comanda **DO CASE**. Dacă nicio condiție nu este adevărată, se va executa secvența <secvm> asociată clauzei **OTHERWISE**.

Structura repetitivă și comanda **DO WHILE . . .ENDDO**

```
DO WHILE <cond>
  <secv >
  [LOOP]
  [EXIT]
ENDDO
```

Structura repetitivă condiționată anterior (sau structura **while**) se execută astfel: câtă vreme condiția pusă în structură <cond> este adevărată, se repetă grupul de comenzi <secv>. În momentul în care condiția a devenit falsă, se părăsește ciclul.

Structura repetitivă și comanda **SCAN . . . ENDSCAN**

```
SCAN [<domeniu>]
[FOR <cond>]
[WHILE <cond>]

    <secv>

ENDSCAN
```

Se parcurge baza de date activă, selectând articolele prin clauzele <domeniu> **FOR**<conditie> și **WHILE** <conditie>. Secvența de comenzi <secv> se va aplica pe articolele selectate. Domeniul implicit de acțiune al comenzii **SCAN** este **ALL**. Se poate forța ieșirea din structura repetitivă **SCAN** prin comenzile **EXIT** și **LOOP**.

Structura repetitivă și comanda **FOR . . . ENDFOR**

```
FOR <variabila> = <exp1>
TO <exp2> [STEP <exp3>]

    <secv >

ENDFOR
```

Se inițializează variabila de control a ciclului cu valoarea <exp1>; se execută corpul de instrucțiuni până când variabila de control devine mai mare decât <exp2>. La fiecare pas se modifică valoarea variabilei de control a ciclului cu valoarea <exp3>, dacă este prezentă clauza **STEP**, sau incrementată cu o unitate dacă lipsește clauza **STEP**.

Ieșiri forțate: **LOOP** și **EXIT**

Pentru forțarea ieșirii dintr-un ciclu se folosește comanda **LOOP**, care determină saltul peste următoarele instrucțiuni ale ciclului și reevaluarea condiției ciclului. **EXIT** determină ieșirea forțată din buclă, indiferent de valoarea expresiei logice <cond>.

Exemple

1. Dorim să introducem într-o tabelă numele și conținutul mai multor documente pe care le avem într-un director. Vom folosi câmpul General.

```
CREATE TABLE Fisier (nume C(24),text G)
CD GETDIR()
Nr = ADIR(A, "*.doc")
IF nr > 0
    FOR i = 1 to nr
        APPEND BLANK
        REPLACE fisier.Nume WITH A(i,1)
        APPEND GENERAL fisier.text FROM A(i,1)
    ENDFOR
ELSE
    =MESSAGEBOX("Nu sunt documente!")
ENDIF
```

Crearea tabelii; poziționare pe director prin GETDIR. Funcția ADIR copiază într-un masiv informațiile despre fișierele din directorul curent. NR=Număr de fișiere copiate în matricea A. Dacă avem documente, atunci adăugăm câte un articol în fișier, reținând numele și conținutul documentului.

2. Utilitatea funcției GetFile

```
SELECT 0
FIS = GETFILE('DBF')
DO CASE
    CASE 'UNTITLED' $ FIS
        CREATE (FIS)
    CASE EMPTY (FIS)
        RETURN
    OTHERWISE
        USE (FIS)
        BROWSE
    ENDCASE
```

Se deschide fereastra Open.

Opțiuni:

1. Dacă a fost apăsat butonul New, atunci șirul returnat are valoarea 'Untitled' și se permite crearea tabelii;
2. Dacă a fost apăsată tasta Cancel sau Esc sau butonul Close, atunci șirul returnat este vid și nu facem nimic;
3. A fost selectat fișierul și acesta va fi deschis prin comanda Use.

3. Problema celebrității

Spunem că o persoană este **celebritate** pentru un grup, dacă este cunoscută de toți membrii grupului, dar nu cunoaște pe nimeni. Stabiliți cine este celebritatea (dacă există o astfel de persoană).

Rezolvare

Presupunem existentă o tabelă RELATII(P1 N(4), P2 N(4)) cu semnificația „persoana de cod P1 cunoaște persoana de cod P2“.

P1 și P2 conțin coduri de persoane (valori numerice întregi mai mari ca 1). Valoarea maximă a acestor coduri poate da numărul total de persoane.

1) Determinarea codului maxim în variabila **nr**

Varianta 1

```
Use relatii
Do while not eof()
if nr<max(p1,p2)
    nr=max(p1,p2)
```

```
skip
enddo
endscan
```

2) Afișarea celebrității

```
Index on p1 tag p1
for i=1 to nr
y=0
scan for p2=i &&numarare
y=y+1
endscan
if y=nr-1 &&pers i este
&&cunoscuta
seek i
if .not. found() &&nu cunoaste
?'persoana',i,'e celebritate'
exit && iesire foretata
endif
endif
endfor
use
```

Varianta 2

```
calc max(p1), max(p2) to n, m
nr=max(n, m)
```

```
Mesaj =' ' && initial, sirul vid
For i=1 to Nr
count for p2=i to y
if y=nr-1
locate for p1=I
if .not. found()
mesaj="pers"+str(i)+;
'este calebritate'
EXIT
Endif
endif
endfor
if empty(mesaj)
? 'nu exista celebritate'
else
? mesaj
endif
```

4. O problemă de vânzare a biletelor de tren

La agenția CFR, pentru activitatea de eliberare a biletelor, este folosit un fișier LOCURI cu următoarea structură (figura 12-1):

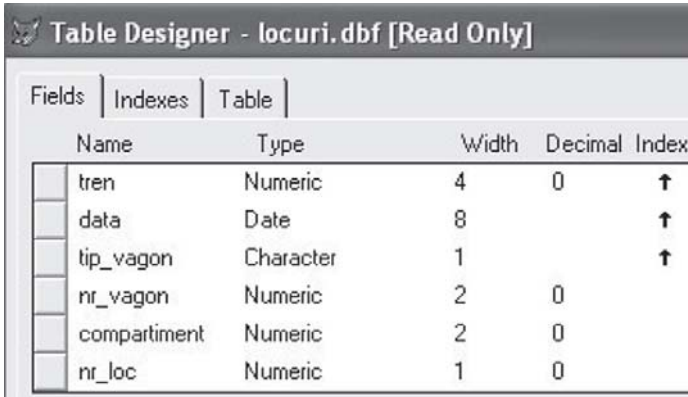


Table Designer - locuri.dbf [Read Only]

Fields	Indexes	Table			
Name	Type	Width	Decimal	Index	
tren	Numeric	4	0	↑	
data	Date	8		↑	
tip_vagon	Character	1		↑	
nr_vagon	Numeric	2	0		
compartiment	Numeric	2	0		
nr_loc	Numeric	1	0		

La sosirea unui client se cere data, numărul trenului, tipul vagonului (clasa 1, clasa 2, cușetă sau vagon de dormit) și numărul de bilete solicitate. Biletele se pot elibera maxim cu zece zile înaintea plecării trenului.

Figura 12-1:
Fișierul locuri.dbf
în Table Designer

Vanzare.prg

```
use locuri in 1
input 'trenul' to t
input 'nr-bilete?' to n
accept 'data?' to d
d=ctod(d)
accept "clasa? (1,2,d,c)" to c
sele 1
set filter to a.tren=t and a.data=d and a.tip_vagon=c
count to v
if v<n
messagebox("atentie, nu sunt bilete")
else
locate for tren=t and data=d and tip_vagon=c and not ocupat
i=0
do while found() and i<n
i=i+1
@ 5+i,5 say "aveti locul"+str(nr_vagon)+str(compartiment)+str(nr_loc)
replace ocupat with .T.
continue
enddo
```



sarcini de laborator

1. Cum modificați procedura de ocupare a locurilor dacă s-ar cere ca toate biletele unui grup să fie în același compartiment sau același vagon? Este necesară precizarea stației de coborâre? Cum s-ar schimba programul dacă ar fi o bază de date distribuită pe o rețea de calculatoare și toate agențiile CFR din țară ar avea acces direct la ea?
2. Scrieți un program care să calculeze numărul de locuri ocupate și neocupate, pe categorii de vagoane și date calendaristice. Aflați care este ziua/zilele când au fost ocupate integral locurile unui tren dat.

3.¹ Fie tabela meciuri, cu rezultatele meciurilor planificate într-o etapă MECI(e1, e2, g1, g2). Realizați un clasament pe echipe, după punctaje și golaveraje.

4.² La un concurs de orientare turistică se fixează de către arbitri momente decalate de plecare în cursă pentru fiecare participant. Organizarea participanților se face pe grupe de vârstă și sex. Grupa este un cod format din litera „m” sau „f” și un număr care indică limita maximă a intervalului de vârstă. De exemplu grupa „m10” desemnează băieții între 9 și 10 ani, „f8” fetele între 7 și 8 ani.

În fișierul START (grupa, moment-start, interval) se introduc pentru fiecare grupă momentul de plecare în concurs și intervalul între două plecări succesive ale membrilor aceleiași grupe. În fișierul CONCURENT (grupa, nr, nume, moment-plecare, moment-sosire) se vor trece concurenții.

La *înscrisoare*, pentru fiecare concurent se citesc numele, data nașterii, sexul și se determină grupa. Câmpul Nr va conține numărul curent după ordonarea alfabetică pe grupe. Momentul de plecare va fi calculat în funcție de momentul de start al grupei și de poziția în grupă.

Scrieți un program pentru:

- 1) introducerea concurenților (nume, vârstă, sex);
- 2) codificarea persoanelor (câmpul Nr);
- 3) introducerea momentelor de start pentru fiecare grupă;
- 4) completarea momentului de plecare în cursă a fiecărui participant (atenție la intervalul între plecările celor din aceeași grupă);
- 5) înregistrarea momentului de sosire a fiecărui participant (pentru o persoană indicată prin cod se va trece timpul sistem);
- 6) determinarea câștigătorului din fiecare grupă;
- 7) afișarea listelor de final pe grupe, în funcție de durata cursei;
- 8) afișarea tuturor concurenților în ordine alfabetică.

5. Observați programul următor. Puteți deduce structura tabeli contracte? Ce credeți că dorește să realizeze programul? Ce erori sunt?

```
1. create table manevra(nr_art      11.endif
   N(3), erori N(3))              12.if not inlist(um, 'kg', 'tone',
2. use manevra in 1 alias ma      'buc')
3. use contracte in 2 alias c     13.er=er+str(3)
4. er=[ ]                        14.endif
5. scan                          15.if er # [ ]
6. if empty(furnizor)           16.sele ma
7. er=er+str(1)                 17.repl nr_art with recno(2),
8. endif                        1-> erori with m->er
9. if not between(term_livr,    18. endscan
   date(), date()-7)
10.er=er+str(2)
```

¹ Urmăriți o sugestie de rezolvare la sfârșitul manualului.

² Urmăriți o sugestie de rezolvare la sfârșitul manualului.

Proceduri utilizator

Procedurile utilizator sunt unități funcționale care realizează o anumită sarcină și returnează sau nu valori în modulul apelant. Pot fi plasate în fișierul sursă în care se găsește modulul apelant sau în alt fișier, numit **fișier de proceduri**. Definirea procedurii:

```
PROCEDURE <nume-proc>
[PARAMETERS <lista-parametri -formali>]
<comenzi>
[ENDPROC]
```

Comunicarea între modulul apelant și procedură se poate face prin parametri. Definirea listei de parametri formali se realizează prin comanda **PARAMETERS**, care trebuie să fie pe prima linie după numele modulului.

Apelul unei proceduri se face prin comanda **DO**:

```
DO <nume-proc> [WITH <lista parametri-efectivi>]
```

Lista parametrilor reali sau efectivi este precizată în clauza **WITH** a comenzii de apel.

Revenirea în programul apelant se poate face prin una dintre comenzile:

```
RETURN[TO MASTER]/CANCEL/RETRY
```

Funcții utilizator

Funcția este o unitate funcțională care returnează programului apelant o valoare, ca rezultat al prelucrărilor sale. O funcție utilizator (UDF – **U**ser **D**efined **F**unction) poate intra în componența unei expresii.

Definirea unei funcții se face prin comanda **FUNCTION**:

```
FUNCTION <nume-functie>
[PARAMETERS <lista-parametri-formali>]
<comenzi>
RETURN <expr>
[ENDFUNC]
```

Comunicarea rezultatului funcției se face comanda **RETURN <exp>**.

Apelul pentru execuția funcției se face prin numele acesteia în cadrul unei expresii. La execuție, în locul identificatorului se va introduce valoarea returnată de funcție.

Transmiterea parametrilor se poate face prin valoare sau prin referință. În mod implicit, parametrii sunt transmiși prin valoare în funcție și prin referință în subprograme sau proceduri. Schimbarea modulului de transmitere a parametrilor la funcții utilizator se poate realiza prin comanda:

```
SET UDFPARMS TO VALUE / REFERENCE
```


În Visual FoxPro nu există nici o diferență între unitățile funcționale. O procedură poate fi apelată ca o funcție și invers. Numărul de parametri reali poate să fie mai mare decât cei formali! În continuare sunt prezentate câteva unități funcționale. Observați apelurile:

<pre> Procedure ex_proc Parameters unu, doi If parameters()<2 =messagebox("prea putini parametri") Return .F. else Return .T. Endif Endproc </pre>	<pre> Do ex_proc with 15 Do ex_proc with 15,20,30 ? ex_proc(15,20) =ex_proc (15) </pre>
<pre> function ex_proc (unu,doi) If parameters()<2 =messagebox("prea putini parametri") Return .F. else Return .T. Endif Endfunc </pre>	<pre> procedure ex_proc (unu,doi) If parameters()<2 =messagebox("prea putini parametri") Return .F. else Return .T. Endif </pre>

Exemplu

<pre> function adun parameters x x=x+1 return x </pre>	<pre> set udfparms to value A=5 ? adun(A) 6 ? A 5 </pre>	<pre> set udfparms to reference A=5 ? adun(A) 6 ? A 6 </pre>
--	--	--

Când un parametru este transmis prin valoare, orice modificare în unitatea funcțională nu este vizibilă în exterior.

Când un parametru este transmis prin referință, orice modificare a conținutului său este vizibilă în programul apelant.

Exemple

1. Criptare și decriptare. Presupunem că în câmpul Adresa (de tip Memo) al unui fișier se găsesc informații secrete. Se dorește criptarea și, desigur, decriptarea lor.

ELEVI (nume C(10), adresa M)

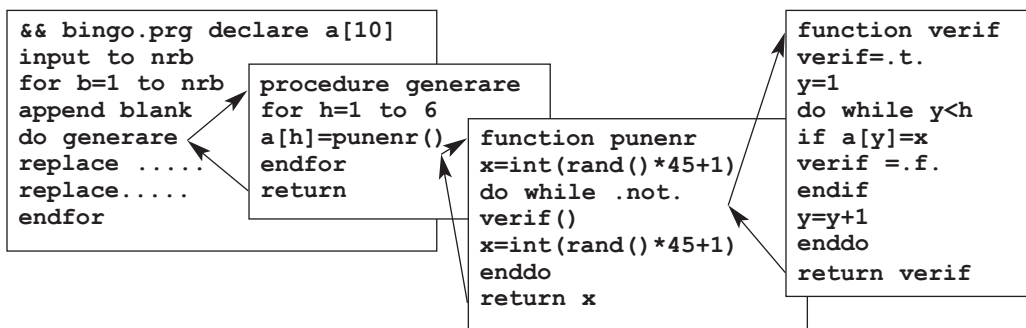
<pre> && criptare replace all adresa; with cripto(adresa) </pre>	<pre> && decriptare Scan wsir='' do decripto with elevi.adresa, wsir replace adresa with wsir endscan </pre>
<pre> function cripto parameters s w='' for i=1 to len(s) el=substr(s,i,1) if el \$ 'aAeEiIoOuU' el=el+'p'+el && "p"+vocala endif w=w+el endfor return w </pre>	<pre> procedure decripto parameters s,w w='' i=1 do while i<= len(s) el=substr(s,i,1) if upper(el) \$ 'AEIOU' i=i+2 endif w=w+el i=i+1 enddo </pre>

2. La jocul BINGO se pun în vânzare săptămânal un număr de buletine de concurs, număr hotărât de agenție. Pe buletine sunt înscrise câte 6 numere distincte, generate aleator, în intervalul 1-45. Buletinele de concurs sunt numerotate (inseriate). Tabela va fi BINGO cu următoarea structură:

d,8	n,8	n,2	n,2	n,2	n,2	n,2	n,2
-----	-----	-----	-----	-----	-----	-----	-----

```

use bingo
declare a[6]
input 'cate buletine ?';
  to nrb
input 'seria primului ?';
  to ns
for b =1 to nrb
  append blank
  repl cod with (ns-1)+b, data with date()
do generare
repl from array a fields nr1,nr2,nr3,;
  nr4, nr5, nr6
endfor
use
return
  
```



3. În vederea vânzării biletelor, fișierul LOCURI din aplicația anterioară trebuie completat cu locurile la toate trenurile din perioada următoare (maxim 10 zile). Să presupunem că există un fișier GARNITURA, actualizat de un dispecer, ce oferă informații despre structura fiecărui tren: câte vagoane are, de ce tipuri.

Name	Type	Width	Decimal	Index	NULL
tren	Numeric	5	0		
nr_vag_1	Numeric	2	0		
nr_vag_2	Numeric	2	0		
nr_cusete	Numeric	2	0		
nr_vag_dormit	Numeric	2	0		

Figura 12-2: Fișierul GARNITURA în Table Designer

Scrieți un program care, pe baza unui meniu, să ofere mai multe activități: editarea tabelii Garnitura, generarea locurilor, vânzarea biletelor. Observați o modalitate de realizare. Atenție la folosirea procedurilor cu parametri!

Programul principal CFR.prg

```
use locuri in 1
use garnitura in 2
opt =3
do while opt# 4
@ 1,1 say '1. editare garnituri'
@ 2,1 say '2. generare locuri'
@ 3,1 say '3. vanzare bilete'
@ 4,1 say '4. terminare program'
@ 5,1 say 'alegeti ' get opt
read
do case
case opt=1
sele 2
browse
case opt=2
do generare
case opt=3
do vanzare
endcase
enddo
close all
```

Procedure generare

```
sele 2
scan
do gen_loc with "1",b.nr_vag_1,8,6
do gen_loc with
"2",b.nr_vag_2,10,8
do gen_loc with
"c",b.nr_cusete,6,6
do gen_loc with
"d",nr_vag_dormit,6,3
endscan
close all
return
```

Procedure gen_loc

```
parameters t,v,c,l
for i=1 to v
for j=1 to c
for k=1 to l
sele 1
append blank
replace id with recno()
replace a.tren with b.tren
replace data with date()+10
replace tip_vagon with t
replace nr_vagon with i
replace compartiment with j
replace nr_loc with k
endfor
endfor
endfor
sele 2
return
```



sarcini de laborator

I. Un **concurs de frumusețe**³ conține mai multe probe sau criterii de selecție. Fiecare criteriu are un punctaj minim trecut în vectorul P din fișierul MINIME.MEM. La înscrierea concurenților, se trec datele generale în fișierul CONCURS.DBF (nume, adresa etc). Odată cu desfășurarea probelor se înregistrează punctajele obținute efectiv de candidate în fișierul RESULT.DBF (nume, criteriu, punctaj). Cerințe:

1. Scrieți o funcție care testează dacă fișierele aplicației există și dacă toate rezultatele au fost completate în fișierul **Result.dbf**. Funcția va determina continuarea sau părăsirea programului.
2. Creați o procedură care compune un tablou cu rezultatele fiecărei concurente pe criterii și afișează tabloul.
3. Scrieți o funcție care determină punctajul maxim obținut de participante la un criteriu dat ca parametru.

³ urmăriți variantele de rezolvare a problemei la sfârșitul manualului.

4. Scrieți o procedură care va marca pentru ștergere toate persoanele din **Rezult.dbf** care nu au obținut minimul cerut de fiecare criteriu și afișează aceste persoane.
5. Scrieți o procedură care afișează persoanele câștigătoare la fiecare criteriu.
6. Scrieți o funcție care testează dacă există criterii la care există mai mulți câștigători și returnează codurile acestor criterii.
7. Scrieți o funcție care află numărul persoanelor cu punctaj maxim la vreun criteriu și sub minim la altul. Care este (sunt) acest(e) caz(uri)?
8. Scrieți o procedură care afișează clasamentul pe primele trei locuri. Participă toate concurențele care au punctaje de trecere la toate probele. La același punctaj se acordă aceleași medalie.
9. Scrieți o procedură care permite înregistrarea punctajelor obținute la o probă (criteriu) dată ca parametru la toate concurențele.
10. Scrieți o procedură care citește datele personale ale unei concurențe.
11. Scrieți o procedură de actualizare a vectorului cu punctaje.
12. Scrieți programul principal care apelează aceste unități funcționale.

II. Fie următoarele fișiere. Presupunem că un programator distrat a construit un set de variante de fișiere principale și un alt set de fișiere secundare, adică cele care conțin unitățile funcționale apelate de principal.

```
**fișier1.prg
procedure unu
parameters a
?a*10
return
function doi
? a/10
return
```

```
**fișier2.prg
clear
do unu
? doi
return
```

```
**fișier3.prg
function unu
parameters a
return a*10
procedure doi
parameters a
return a/10
```

```
**fișier4.prg
clear
do unu(5)
do doi
return
```

```
**fișier5.prg
procedure unu
parameters a
?a*10
function doi
?a/10
return .t.
```

```
**fișier6.prg
clear
? unu(5)
?doi(5)
```

Care este perechea corectă?



sarcini pentru realizarea unui mini-proiect

Identificați prelucrările specifice fiecărui domeniu analizat și proiectați procedurile necesare. De exemplu, pentru baza de date TEATRU vă propunem să analizați următoarele proceduri:

1. O procedură de planificare a unui spectacol pentru o anumită dată/sală/oră – numai dacă sala nu este ocupată;
2. O procedură de afișare a informațiilor solicitate de clienți (de exemplu, unde se joacă piesa x? La ce oră încep spectacolele la sala x? etc.);

3. O procedură de rezervare/vânzare a biletelor la spectacolele din orașul X;
4. O procedură pentru calcule statistice: număr de spectatori, sume încasate, valoarea medie a unui bilet etc.

Activitatea de depanare a programelor

Testarea sau depanarea dezvăluie deficiențele de proiectare și erorile de scriere, ceea ce înseamnă pentru programator revenirea la faza de codificare, corectarea erorilor, recompilare, apoi relansarea în execuție pentru testare.

Primul caz de depanare a avut loc cu mulți ani în urmă, când o insectă a provocat defectarea câtorva componente din calculator. A apărut termenul de **debugging**=depanare, adică operația de scoatere a hibelor (**bugs=insecte**) dintr-un program.

Aplicația informatică trebuie privită ca un ansamblu de componente care, chiar dacă au fost proiectate vizual și suntem siguri că sunt corecte, trebuie asamblate și verificate în interdependențele dintre ele.

Activitatea de testare este foarte importantă și este văzută ca o activitate distinctă a procesului de dezvoltare. Se poate desfășura *conform unui plan*. Unii programatori testează aplicația doar la final. Alții preferă ca pe parcursul proiectării să pună și problema testării. Există două abordări ale procesului de testare:

- a) orientată spre date: nu cunoaștem modul de funcționare a programului, dar selectăm o gamă largă de date de test și rulăm programul cu aceste date, observând dacă sunt obținute rezultatele (ieșirile) dorite.
- b) orientată spre funcția programului: cunoaștem bine funcția realizată de program și urmărim trecerea controlului pe toate căile pe care poate evolua execuția programului.

Etapele activității de testare

a. Pregătirea programului pentru testare:

- verificarea manuală a modului de codificare a programului, urmărind respectarea restricțiilor metodologice, alegerea notațiilor standardizate sau semnificative, adoptarea unor formulări clare, precise, utilizarea comentariilor;
- rularea pe calculator a programelor pentru depistarea și corectarea erorilor de compilare;
- adăugarea unor secvențe temporare, care permit controlul derulării programului (mesaje suplimentare, stocarea rezultatelor intermediare într-un fișier de control etc.);
- simularea modulelor lipsă din cadrul structurii programului, fie prin folosirea modulelor vide (nu execută nici un fel de prelucrare, predând controlul programului apelant imediat ce au fost apelate), fie prin module de substituție (module simple cu caracter provizoriu care simulează funcția modulului respectiv).

- b. **Pregătirea datelor de test** are o deosebită importanță, de această operație depinzând în bună parte reușita testării. Trebuie testate toate ramificațiile din program și toate combinațiile posibile ale acestora. Există o serie de tehnici care permit generarea fișierelor de test sau programatorul trebuie să compună fișiere cu datele dorite.

- c. **Testarea propriu-zisă**, adică rularea programului cu datele pregătite pentru test și verificarea rezultatelor obținute prin analiza fișierelor obținute după prelucrare. Pentru analiza rezultatelor se pleacă de la aceleași date și se efectuează manual calculele, simulând funcția programului, pentru a fi comparate cu ieșirile din program.
- d. **Testarea de ansamblu** vizează asamblarea componentelor și constituirea datelor de test pentru toate funcțiunile modulelor programului, testarea propriu-zisă a programului principal și comunicarea către șeful de proiect a momentului de terminare a etapei.

Tipuri de erori

- a) **Erori de sintaxă**. Sunt depistate la compilarea programului. Sistemul afișează într-o fereastră de mesaje eroarea întâlnită.
- b) **Erori logice**. Sunt mai greu de depistat sau de remediat. Nu sunt detectate la compilare, ci la execuție. De exemplu:
- operația de restaurare **Restore**, chiar dacă folosim clauza **Additive**, poate determina ștergerea vechilor variabile, dacă au același nume;
 - depășirea capacității unui câmp numeric conduce la pierderea valorilor (observați stelutele!);
 - expresii complicate, unde au fost inversate valorile etc.;
 - altă eroare greu de detectat survine când lucrăm cu mai multe zone și pointerul de fișier nu este bine gestionat.

c) **Erori de tip excepție**. Sunt acele situații care nu pot fi anticipate la testarea programului. De exemplu, mutarea unui fișier în alt director sau ștergerea unui index care se face în afara aplicației pot conduce la părăsirea pe caz de eroare a programului.

Sfaturi pentru depistarea și eliminarea erorilor

- Utilizarea modularizării, pentru ca depistarea și corectarea erorilor să se facă mai ușor;
- Eliminarea ieșirilor forțate din structurile de control;
- Conceperea unui plan de testare care să acopere toate căile/ramurile programului;
- Utilizarea ferestrelor **wait** și a altor mesaje de testare a locului unde se găsește execuția programului;
- Crearea unui mediu de testare identic cu cel în care va funcționa aplicația;
- Identificarea operațiilor mai delicate și a calculelor speciale pe care le face programul, pentru a insista supra lor;
- Identificarea criteriilor de succes al testării;

O idee: *Luați programul care nu merge și plasați pe rând proceduri între comentarii până când acesta va funcționa. Refaceți invers drumul, activând câte o procedură până depistați eroarea!*

- **Utilizarea rutinelor (subprogramelelor) de tratare a erorilor.** Rutinele de tratare a erorilor presupun din partea programatorilor un efort de imaginare a eventualelor situații care pot duce la erori și ieșire forțată din program.

Exemple

1. De multe ori, deschiderea unui fișier nu este posibilă din cauza inexistenței lui. În asemenea cazuri, sistemul afișează un mesaj și întrerupe programul. Pentru rezolvarea erorii putem concepe un modul special:

<pre>If ! file ("elevi.dbf") do rezolv endif use elevi</pre>		<pre>Rezolv.prg create elevi use return</pre>
--	--	---

2. Pentru situația detectării sfârșitului de fișier la o comandă **SKIP** se poate construi o procedură care face poziționarea pe ultimul articol. Pentru a elimina dependența de fișierul care a cauzat eroarea vom folosi comanda **ON ERROR**.

<pre>on error do rezolv use elevi skip ...</pre>		<pre>rezolv.prg if error()=4 && codul de eroare este 4 go bottom endif return</pre>
--	--	---

Comanda **ON ERROR** redirecționează execuția către o anumită subrutină când procesorul de evenimente detectează o eroare.

Alte funcții utile

error() && returnează codul de eroare
message() && returnează mesajul asociat erorii
lineno() && returnează numărul liniei care a generat eroarea
program() && returnează numele programului aflat în execuție

Putem realiza o subrutină care să testeze numărul erorii și să încerce să o trateze. De obicei, rutina de tratare a erorilor este apelată cu mai mulți parametri:

```
ON ERROR DO tratare_eroari WITH error(), message(),
program(), lineno()
```

Exemplu

PROCEDURE tratare_eroari

Parameters nreroare, mesaj, Cprogram,nlinie

<pre>Do case case nreroare=1 noufis=getfile('dbf', 'alegeti un fisier') if empty(noufis) cancel</pre>	<p>nu se găsește tabela în directorul precizat</p> <p>vom deschide ecranul de selecție a fișierelor</p> <p>părăsirea programului în caz de eroare</p>
---	---

<code>else</code> <code>return</code>	ne întoarcem în programul apelant după selecția fișierului.
<code>case nreroare=38</code> <code>goto top</code>	pointerul este înaintea primului articol
<code>case nreroare=4</code> <code>go bottom</code> <code>endcase</code>	pointerul este după ultimul articol

Strategii de depanare

Când survine o eroare în timpul execuției programului, sistemul afișează o casetă de dialog conținând mesajul și trei butoane: **Ignore**, **Suspend**, **Cancel**.

- **Ignore** – ignoră eroarea și continuă programul (este cazul unor erori minore, cum ar fi inexistența unui set de culori).
- **Suspend** – suspendă execuția, lasă fereastra de editare deschisă și cursorul plasat pe linia greșită. Mediul este lăsat intact. Operatorul încearcă corectarea și se revine cu comanda **RESUME** din fereastra de comenzi. Aceasta reexecută linia de la care a apărut eroarea. Este cazul unei erori mai grave, cum ar fi lipsa fișierului care trebuie deschis, dar care poate fi corectată manual, direct. Poate fi executată o comandă de creare a unei tabele și apoi se poate reveni. Se pot deschide ferestrele de depanare Trace, Debug.
- **Cancel** – anulează execuția programului și vă plasează în fereastra de editare pe linia greșită.

Utilizarea instrumentului Debugger

Utilitarul **Debugger** se apelează selectând **Tools**, **Debugger** sau prin comanda **SET STEP ON** din fereastra de comenzi. Se deschide o fereastră cu un meniu principal, o bară cu butoane pentru selectarea directă a unor opțiuni, cinci ferestre de lucru, fiecare cu meniuri contextuale.

Principalele ferestre sunt:

Fereastra Trace permite vizualizarea liniilor pe parcursul execuției programului. Deschiderea unui program pentru vizualizare se face prin **File**, **Open** sau prin clic pe butonul cu același nume. Programul deschis trebuie să fie rulat în regim de depanare.

Fereastra Watch ajută la urmărirea valorilor variabilelor, expresiilor. Adăugarea rapidă a unei variabile se face prin poziționarea pe variabilă în fereastra Trace și tragere și plasare spre fereastra Watch.

Fereastra Call stack oferă vizualizarea stivei de apeluri a funcțiilor și procedurilor folosite în program.

Fereastra Output este o fereastră specială în care sunt afișate ieșirile special destinate depanării, prin intermediul comenzii **DebugOut**.

Fereastra Local permite urmărirea variabilelor locale unui modul.

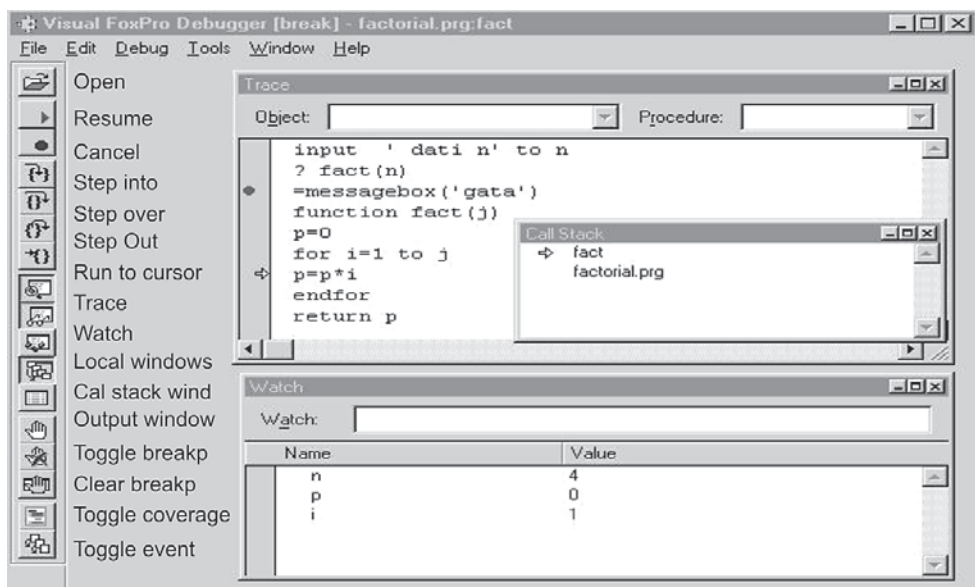


Figura 12-3: Depanarea unui program

Urmărirea execuției unui program se poate face prin rularea acestuia linie cu linie și observarea efectelor asupra variabilelor, câmpurilor, tabelelor. Compararea valorilor expresiilor cu cele așteptate poate să conducă la depistarea erorii. Dacă depistăm o eroare și închidem fereastra depanatorului, intrăm în editarea sursei și facem modificarea.

Opțiunile de rulare a unui program în regim de depanare sunt:

- opțiunea **step into** – rularea pas cu pas intrând și în subrutine. Permite avansul la următoarea instrucțiune numai dacă utilizatorul apasă tasta Enter. Observați în figura 12-3 semnul săgeată către instrucțiunea curentă.
- opțiunea **step over** – rularea pas cu pas, dar sărind peste subrutine. Opțiunea **step out** este folosită dacă, indiferent pe ce instrucțiune am fi, dorim terminarea programului într-un singur pas.
- opțiunea **run to cursor** – executarea tuturor instrucțiunilor de la cea curentă (marcată prin săgeată) într-un singur pas până la instrucțiunea pe care am pus cursorul.
- opțiunea **de rulare încetinită** se alege selectând **Debug, Throttle**. După un anumit număr de secunde, precizat de utilizator, se trece la instrucțiunea următoare.
- opțiunea **breakpoint** – rulare până se ajunge la un punct de întrerupere.

Punctele de întrerupere se plasează prin dublu clic pe banda din stânga a liniei, cu instrucțiunea sau expresia unde se dorește oprirea sau prin butonul **toggle breakpoint** (observați un punct roșu!). Ștergerea unui punct de întrerupere se face prin dublu clic pe el sau cu opțiunea Clear Breakpoint.

Atenție! Un punct de întrerupere poate fi permanent chiar dacă închideți programul și ieșiți din depanare.

Atunci când fixăm un punct de întrerupere este bine să ne gândim ce facilități permite acesta. Fereastra Breakpoints, deschisă selectând **Tools, Breakpoints**, arată aceste posibilități la lista Type:

- a) Oprire la un anumit modul / linie (**Break at location**).
- b) Oprire atunci când o anumită expresie este adevărată sau doar se modifică (**Break when expression is true/Break when expression has changed**). Expresia ce trebuie construită în aceeași fereastră a utilitarului se va evalua la fiecare linie a programului, determinând oprirea atunci când este adevărată sau numai și-a schimbat valoarea.
- c) Oprire la o anumită linie numai dacă expresia este adevărată (**Break at location when expression is true**) este combinația celor două posibilități anterioare.

Ori de câte ori programul întâlnește un punct de întrerupere, îl tratează ca pe o instrucțiune Suspend și întrerupe execuția. În timp ce programul este suspendat puteți face următoarele:

- a) să parcurgeți codul sursă înainte și înapoi folosind bara de derulare;
- b) să comutați la un alt obiect sau altă procedură;
- c) să deschideți fereastra Data Session pentru a verifica starea fișierelor deschise, indecșii asociați;
- d) să introduceți comenzi în fereastra de comenzi;
- e) să selectați opțiuni din meniul principal;
- f) să modificați valoarea curentă a oricărei variabile de memorie sau tabel.



sarcini de laborator

I. Fie un program de calcul a factorialului. Programul citește o variabilă, apelează funcția de calcul și la terminare afișează un mesaj. Observați-l în fereastra **Trace** din figura 12-4. De fiecare dată când îl rulăm obținem același rezultat: zero!

1. Scrieți codul programului de calcul al factorialului – exact ca și cel din figură – cu editorul de programme.
2. Apelați programul utilitar prin **Tools, Debugger**.
3. Deschideți fereastra Trace prin clic pe butonul **Trace**.
4. Deschideți programul în fereastra de urmărire prin **File, Open** sau clic pe butonul cu același nume etc.

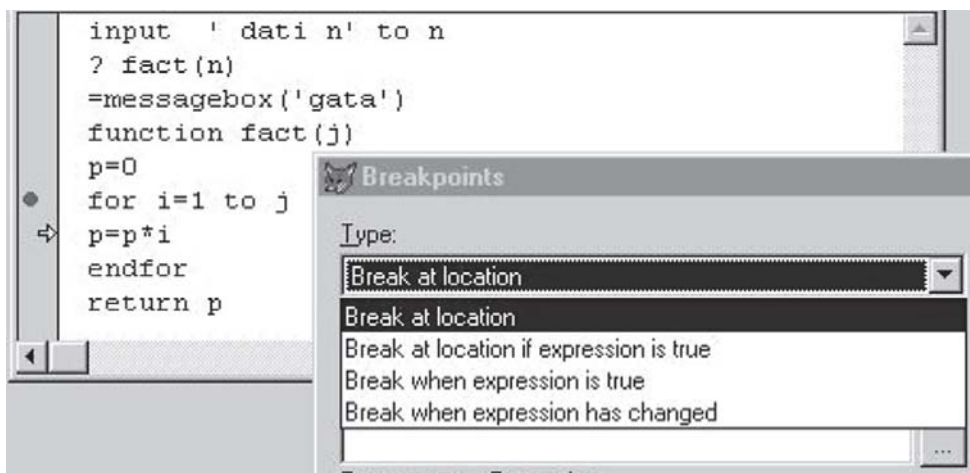


Figura 12-4: Exemplu de întrerupere

5. Deschideți fereastra Watch prin **Window, Watch** sau clic pe butonul utilitar.
6. Treceți în urmărire variabilele de lucru: n, i, p prin tragere și plasare.
7. Apăsăți butonul **Step Over**. Ce se întâmplă? Programul se va opri la introducerea lui n și va afișa 'gata!'. Valoarea rezultatului este zero! Mai mult, se șterge sursa din fereastra Trace.
8. Redeschideți sursa și rulați-o pas cu pas prin butonul **Step Into**. Observați deplasarea săgeții de la o instrucțiune la alta și modul de variație a variabilelor i și p. Ați depistat eroarea? Dacă nu, continuați depanarea.
9. Fixați un punct de întrerupere pe instrucțiunea **FOR** și activați butonul **Step over**. Automat programul trece peste celelalte comenzi și se oprește doar la **FOR**. Observați în fereastra Watch valoarea lui i și a lui p. Continuați alt **Step over**. Ați observat sursa erorii? Ce s-a întâmplat cu p? Care este ultima valoare a lui i (când iese din **FOR**)?
9. Deschideți fereastra Breakpoint și observați ce acțiune este implicită la fixarea punctului de întrerupere pe o linie de comandă!
10. Ștergeți punctul de întrerupere de pe linia **FOR** și puneți unul pe variabila p în fereastra Watch. Apăsăți butonul **Step over**. De câte ori s-a oprit programul? Niciodată? De ce?
11. Puneți un alt punct de oprire pe linia p=p*i din subrutină și plasați în fereastra Breakpoint opțiunea Break at line when changed. Executați rularea pas cu pas. De câte ori s-a oprit programul? Niciodată? De ce?
12. Ce trebuie făcut pentru a modifica sursa? Comanda Modify Command este suficientă?

II. Am realizat un mic program care numără contractele pe beneficiari. Apare în fereastra 2.

1. Datele de plecare sunt afișate în fereastra 1.

Contracte					
Nr_contr	Data	Nume_benef	Nume_prod	Pret_promi	Cant
122	09/23/99	roman	pantofi	200000	200
123	07/07/89	suceava	papuci	70000	130
124	07/08/98	alba	pantofi	209000	300
125	01/09/98	alba	sandale	100000	200
600	09/08/99	roman	pantofi	100000	1000

Figura 12-5: Fereastra 1

Deși **pare ciudat**, rezultatul afișat pe ecran ca urmare a execuției programului a fost următorul:

```
numarul contractelor beneficiaruluiroman este 1
numarul contractelor beneficiaruluialba este 2
```

Unde am greșit?

2. Scrieți ce acțiuni au fost executate pentru a obține fiecare dintre cele patru ferestre!

```
use contracte
scan
numeben=nume_benef
count while nume_benef=numeben to x
?' numarul contractelor beneficiarului'+numeben
??'este ', x
endscan
use
```

Fereastra 2

Name	Value
numeben	"roman "
nume_benef	"roman "
recno()	1
numeben=nume_benef	.T.

Fereastra 3

Figura 12-6:
Ferestrele 2 și 3

3. Unde s-a fixat punctul de întrerupere, pe o variabilă sau pe o instrucțiune?

Figura 12-7: Fereastra 4

Operații cu baze de date în Visual FoxPro

- Comenzi și manipulare interactivă a bazelor de date
- Comenzi și manipulare interactivă a tabelelor incluse
- Stabilirea legăturilor între tabele
- Proceduri stocate și declanșatoare

O **bază de date** este o colecție de date operaționale, cu o anumită importanță și stabilitate în domeniul supus analizei și proiectării unei aplicații informatice. Baza de date relațională nu poate fi privită doar ca un ansamblu de tabele, ci cuprinde și relațiile dintre acestea, relații permanente (nu cele definite în programele utilizator pentru o anumită prelucrare), condițiile care trebuie îndeplinite pentru a asigura corectitudinea și integritatea colecției în întregime și anumite informații¹ – cum ar fi numele lung al câmpurilor, formatul implicit de afișare, valorile implicite ale câmpurilor, funcții pentru validarea câmpurilor sau a înregistrărilor etc.

Manipularea bazei de date în SGBDR Visual FoxPro se poate face atât prin comenzi, cât și în mod interactiv, Visual FoxPro punând la dispoziție aplicații wizard (vrăjitori) și aplicații Designer (constructori).

Descrierea operațiilor cu baze de date prin comenzi

Vom preciza principalele operații cu o bază de date și comenzile necesare.

CREATE DATABASE <fis.dbc>	Crearea unei noi baze de date – fișier cu extensia .DBC. Numai baza de date nou-creată este deschisă automat.
OPEN DATABASE <fis.dbc>	Deschiderea bazei de date specificate pentru a avea acces la conținutul ei – chiar dacă toate tabelele și vizualizările sunt disponibile, nu înseamnă că sunt deschise, trebuind executate acțiuni explicite pentru deschiderea lor.
SET DATABASE TO <fis.dbc>	Fixarea bazei de date curente dintre cele deschise – la un moment dat, o singură bază de date este activă și o vom numi baza curentă.
CLOSE DATABASES/ALL	Închiderea bazei de date curente, împreună cu tabelele ei – opțiunea ALL închide bazele de date și tabelele lor, tabelele izolate (FREE), indecșii și fișierele format.

¹ incluse într-un Dicționar de date care asigură – așa cum am prezentat în aspectele teoretice – independența sau autonomia datelor față de programe.

DELETE DATABASE <dbf> [DELETETABLES]	Ștergerea bazei de date specificate de pe disc – ea nu trebuie să fie deschisă. Dacă opțiunea DELETETABLES nu este prezentă, tabelele conținute în baza de date nu sunt șterse de pe disc, ci numai din bază.
ADD TABLE <dbf>	Adăugarea unui fișier .DBF în baza activă
REMOVE TABLE <dbf> [DELETE]	Ștergerea tabelului precizat din baza de date, dar nu și de pe disc – clauza DELETE permite ștergerea și de pe disc a fișierului. Atenție! Tabela trebuie să fi fost închisă anterior comenzii.
DISPLAY DATABASES	Afișarea informațiilor despre conținutul bazei de date.
DISPLAY TABLES	Afișarea numelor tabelurilor existente în baza de date curentă.

Gestiunea interactivă a bazei de date. Database Designer

Utilitarul Database Designer permite definirea tuturor operațiilor cu baza de date prin intermediul meniurilor și al butoanelor. Se lansează astfel:

- pentru crearea bazei de date: din meniul sistem, selectând **File, New, Database, New** sau prin comanda **CREATE DATABASE**.
- pentru modificarea unei baze de date existente: selectând **View, Database Designer** sau prin comanda **MODIFY DATABASE**.

Meniul Database

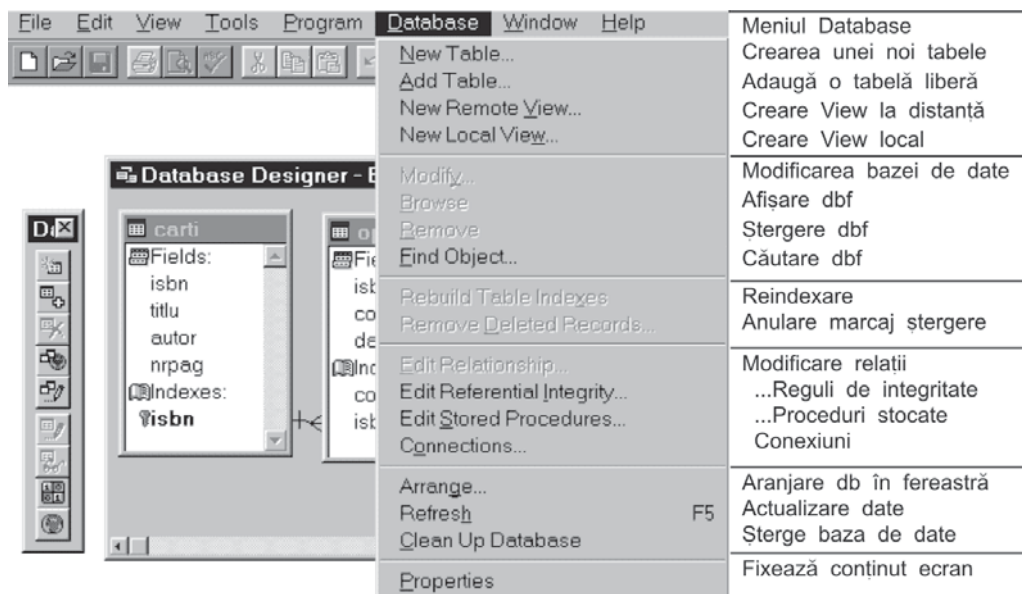


Figura 13-1: Meniul Database

Meniul contextual Database

Utilitarul dispune de un **meniul contextual** care permite un alt mod de selectare a operației dorite prin clic dreapta. În general, opțiunile sunt aceleași ca și cele incluse pe barele de butoane.

În plus, **Expand All** afișează toate tabelele cu structura lor; **Collapse All** afișează numai denumirea tabelor; **Help** deschide fereastra de asistență; **Properties** fixează tipurile de fișiere care vor fi vizualizate în fereastra Database Designer.

Bara de instrumente asociată utilitarului Database Designer se deschide selectând **View, Toolbars**.

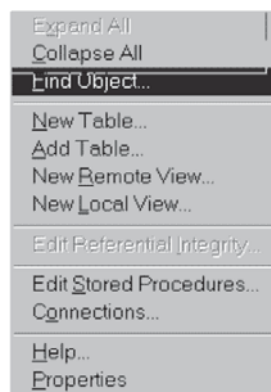


Figura 13-2: Meniul contextual Database

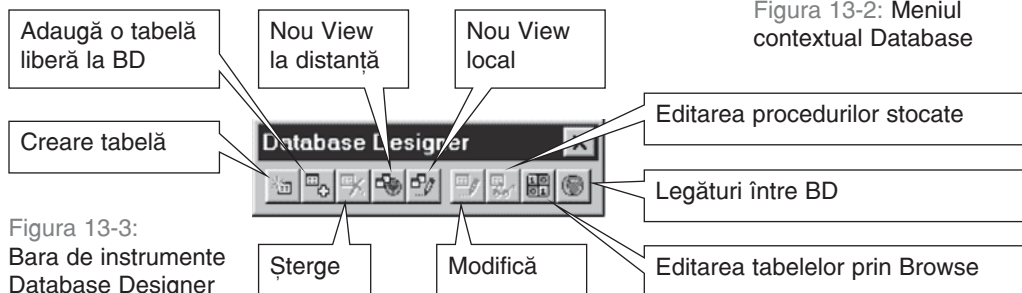
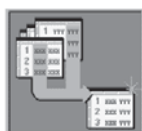


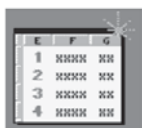
Figura 13-3: Bara de instrumente Database Designer

Crearea rapidă a unei baze de date cu Database Wizard

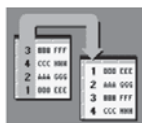
Asistentul Database Wizard se poate apela fie selectând **File, New, Database, Wizard** fie selectând **Tools, Wizards, Database**. Pentru crearea bazei de date cu utilitarul Database Wizard, parcurgeți următorii pași:



Pasul 1: Alegeți una dintre bazele de date disponibile (legate de bibliotecă, studenți, gestiune etc.). Selectați tabela și fișierele view care vor forma noua bază de date.



Pasul 2: Pentru baza de date selectată, păstrați toate tabelele pe care le veți include sau alegeți câteva.



Pasul 3: Selectați câmpurile din fiecare tabelă. Pentru fiecare tabelă vizualizați câmpurile și eventual schimbați cheile de indexare și cheia primară.



Pasul 4: Stabiliți interactiv relațiile dintre tabele: pentru fiecare tabelă precizați care este legătura. Observați că sunt deja fixate legături. Le puteți păstra, șterge sau puteți adăuga altele.



Pasul 5: Salvați definițiile și, eventual, populați tabelele. După salvare, baza de date este generată și puteți lucra cu ea. Atenție, ea nu este automat deschisă.

Deschiderea și activarea unei baze de date

Se poate face prin comanda `OPEN DATABASE ?/<dbc>` în fereastra de comenzi sau selectând **File, Open, Database**. Putem deschide mai multe baze de date. Se pot observa toate bazele de date deschise într-o listă derulantă plasată pe bara de instrumente.

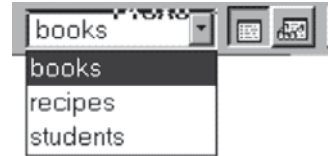


Figura 13-4: Lista derulantă a bazelor de date



sarcini de laborator

1. Creați o bază de date cu numele Bib, preluând baza de date predefinită Books.
2. Observați fișierele componente și aranjarea lor în fereastra Database Designer. Vizualizați conținutul tabelor.
&& ne putem poziționa pe bara de titlu și prin dublu clic deschidem fereastra Browse && sau prin comanda Browse din meniul contextual sau din meniul Table
3. Închideți baza de date.
&& nu uitați de comanda Close database!
4. Închideți fereastra Database și Table Wizard.
5. Creați o altă bază de date cu numele Studenți, însă preluând selectiv unele tabele și relații din baza de date predefinită Students.
6. Deschideți și baza de date Bib alături de Studenți.
&& fileopen
7. Activați când una, când cealaltă bază de date și observați fereastra Database Designer.
&& se poate folosi comanda Modify database
8. Folosiți meniul contextual atașat ferestrei Database Designer pentru micșorarea tuturor tabelor și revenirea lor la dimensiunile proiectate.
9. Ștergeți din baza de date Bib câteva tabele. Ele vor deveni tabele libere. Ștergeți de pe disc câteva tabele. Ștergeți baza de date Bib.



sarcini pentru realizarea unui mini-proiect

Creați baza de date pentru activitatea analizată și adăugați tabelele create deja. Folosiți fereastra Database Designer!

Operații asupra unei tabele incluse în baza de date

În dicționarul bazei de date, așa cum am văzut, se memorează informații suplimentare pentru structura conceptuală la nivelul fiecărei tabele, cum ar fi: numele lung asociat câmpurilor, valoarea implicită, șablonul de afișare, mesajele către utilizator pentru cazuri de eroare, dar și mesaje către proiectant pentru cazuri de restructurare a bazei de date, legăturile dintre tabele, regulile de integritate, cheile de indexare etc.

Principalele operații asupra tabelor incluse în baza de date sunt: creare, actualizare, indexare, filtrare, relaționare. Desigur, sunt valabile și comenzile pe care le-am învățat pentru tabele izolate, dar vom prezenta și altele, care fac parte din limbajul SQL.

1. Proiectarea tabelei prin comanda SQL **CREATE TABLE**

O cale de proiectare a structurii este dată de comanda SQL **CREATE TABLE**.

Comanda **CREATE TABLE**² are forma:

```
CREATE TABLE <fis.dbf> NAME <nume-lung>
(<lista-cmp><clauze-articol>)
```

unde:

<lista-cmp>==<nume-câmp1><tip>[(<lung>[, <zecim>])] <clauze-cmp>, ..
<clauze-cmp> se folosesc la nivelul fiecărui câmp:

NULL/ NOT NULL	Atributul poate conține valori NULL
CHECK <exp> [ERROR <mesaj>]	Funcția de validare cu mesajul adecvat
DEFAULT<exp>	Valoarea implicită a atributului
PRIMARY KEY	Declară atributul cheie primară
REFERENCES <tabela2> [TAG<reper>]	Declară atributul ca fiind cheie străină în tabela curentă și cheie de indexare în <tabela2>

<clauze-articol> se folosesc la nivelul tabelei:

PRIMARY KEY <exp> TAG <tag>	Creează indexul primar prin specificarea expresiei și a numelui de reper.
CHECK <exp> [ERROR <MESAJ>]	Permite specificarea unor restricții la nivel de tabelă.
FOREIGN KEY<exp> TAG <tag1>	Fixează drept index o cheie străină și stabilește o legătură cu tabela <dbf> prin acest reper <tag1> și <tag2> din tabela părinte. Dacă lipsește <tag2>, atunci legătura se face prin cheia primară a <dbf>.
REFERENCES <dbf> [TAG<tag2>]	

2 Este parțial cunoscută din lecțiile despre proiectarea structurii tabelor neincluse într-o bază de date; o vom dezvolta prezentând clauzele necesare definirii structurii unei tabele incluse într-o bază de date.

Exemplu

Fie tabele ELEVI și CLASE legate prin atributul cls. Vom scrie comenzile de creare a acestei baze. Câmpul cod este cheie unică pentru Elevi, iar cls este cheie unică pentru Clase.

```
CREATE DATABASE elevi          && definim baza de date
CREATE TABLE clase(cls C(3) NOT NULL DEFAULT "999" PRIMARY KEY,;
    profil C(10) NULL DEFAULT "Info",;
    diriginte C(30) NULL DEFAULT space(30))
CREATE TABLE elevi(cod N(3) NOT NULL DEFAULT 999 PRIMARY KEY,;
nume C(30) DEFAULT SPACE(30), med N(5,2) default 9.99,;
cls C(3) NOT NULL DEFAULT "12A";
CHECK val(left(codcls),2)>=9 ERROR "Atentie anul",;
FOREIGN KEY cls TAG cls REFERENCE clase)
```

2. Modificarea structurii

Modificarea structurii unei tabele se poate face și printr-o comandă SQL. Sunt mai multe forme ale comenzii pe care le prezentăm pe rând:

```
ALTER TABLE <dbf> ADD <cmp><tip> [( <lung>[, <zecim>])]
    [NULL/NOT NULL]
    [CHECK <expl>[ERROR <msg>]]
    [DEFAULT <exp>]
    [PRIMARY KEY ] [REFERENCES <dbf> [TAG <tag>]]
```

sau

```
ALTER TABLE <dbf> ALTER COLUMN <cmp> [NULL/NOT NULL]
    [SET CHECK <expl>[ERROR <msg>]/ DROP CHECK]
    [SET DEFAULT <exp>]/DROP DEFAULT]
```

sau

```
ALTER TABLE <dbf> DROP COLUMN <cmp>
    [SET CHECK <expl>[ERROR <msg>]/ DROP CHECK]
    [ADD PRIMARY KEY <exp> TAG <tag>/DROP PRIMARY KEY]
    [ADD FOREIGN KEY<exp> TAG<tag> REFERENCES<dbf>[TAG<tag>]]
    /DROP FOREIGN KEY TAG <tag> [SAVE]
    [RENAME COLUMN <cmp_vechi> TO <cmp_nou>]
```

Exemple

Fie comanda de creare a tabelei pentru evidența împrumuturilor de cărți.

```
CREATE TABLE imprumut (inv N(5), data_i D, data_r D)
```

- Se adaugă o coloană.
ALTER TABLE imprumut ADD COLUMN cod_cit C(3) NOT NULL
- Se fixează o cheie primară la tabela creată anterior.
ALTER TABLE imprumut ADD PRIMARY KEY str(inv)+dtos(data_i) TAG in_d
- Se adaugă la un câmp o regulă de validare.
ALTER TABLE imprumut ALTER COLUMN data_i SET CHECK data_i>=data_r
- Se adaugă o relație 1-n între carti și tabela Imprumut.dbf pe cheia Inv.
ALTER TABLE imprumut ADD FOREIGN KEY inv TAG inv REFERENCES carti
- Se șterge regula de validare.
ALTER TABLE imprumut ALTER COLUMN data_i DROP CHECK
- Se șterge relația stabilită anterior.
ALTER TABLE imprumut DROP FOREIGN KEY TAG inv SAVE

3. Închiderea unei tabele

```
CLOSE TABLE <dbf>/TABLES/ALL
```

Închide tabela <dbf> din baza de date curentă, toate tabelele din baza curentă sau toate tabelele libere din toate zonele dacă nu este deschisă nici o bază de date.

4. Redenumirea unei tabele

```
RENAME TABLE <dbf_vechi> TO <dbf_nou>
```

Schimbă numele unei tabele în contextul aceleiași baze de date.

Exemple

```
CREATE DATABASE D1          && Creează d1.dbc
CREATE DATABASE D2          && Creează d2.dbc
SET DATABASE TO D1         && Fixează d1 bază curentă
CREATE TABLE T1 (CMP1 n(3)) && Creează t1.dbf în baza d1
CLOSE TABLES              && Închide fișierul t1.dbf
REMOVE TABLES T1          && Șterge t1 din baza d1
SET DATABASE TO D2         && Fixează d2 bază curentă
ADD TABLE T1              && Aduagă t1.dbf la baza d2
RENAME TABLE T1 TO T2     && Schimbă numele la t2.dbf
```

5. Inserarea datelor

Este o comandă care nu trebuie confundată cu versiunea Fox a acesteia pe care am prezentat-o la lecțiile de actualizare. Ea permite adăugarea unei înregistrări la sfârșitul unei baze de date.

Sintaxa comenzii:

```
INSERT INTO <tabela> [(<lista campuri>)] VALUES (<lista valori>)
```

sau

```
INSERT INTO <tabela> FROM ARRAY <vector> / FROM MEMVAR
```

Comanda nu afectează numărul zonei de lucru curente. Dacă tabela în care dorim să facem inserarea este deschisă în altă zonă de lucru decât cea curentă se va adăuga o înregistrare la sfârșitul său fără ca tabela să fie activă.

Exemple

- Să se adauge un elev:

```
INSERT INTO ELEVI VALUES (100, "POPESCU",9.66, "9A")
INSERT INTO ELEVI( nume, cod, med) VALUES ("ionescu", 200, 10)
```
- Să se adauge primul elev în tabela Bacalaureat cu aceeași structură:

```
Use elevi
Scatter memvar
Insert into bacalaureat from memvar
```

6. Ștergerea datelor

Ștergerea poate fi efectuată și printr-o comandă SQL cu formatul general:

```
DELETE FROM [<nume-baza>!]<nume-tabela> WHERE <cond>
```

Exemplu

- Ștergerea elevilor din clasa "9a":
`DELETE FROM ELEVI WHERE cls="9a"`

7. Actualizarea datelor

O altă comandă pentru corectarea valorilor este comanda SQL. Are formatul general:

```
UPDATE [<nume-baza>!]<nume-tabela> SET <cmp1>=<exp1>,  
<cmp2>=<exp2>, ..WHERE <cond>
```

Exemplu

- Modificarea clasei elevului "Popescu" la valoarea "9x":
`UPDATE ELEVI SET cls="9x" WHERE nume="Popescu"`

Proiectarea interactivă a tabeli prin Table Designer

a. Utilitarul Table Designer

Pentru proiectarea directă a structurii unei tabeli se apelează utilitarul Table Designer, fie din fereastra Database Designer prin clic pe butonul **New**, fie selectând **File, New, Table**.

- **Tab-ul Fields** permite fixarea structurii tabeli:

- numele, tipul și dimensiunea fiecărui câmp (ca în proiectarea tabelilor izolate);
- dacă atributul este index și de ce tip (crescător/descrescător) (Index);
- dacă atributul poate conține valoarea NULL sau e obligatoriu să conțină o valoare;
- Șablonul pentru formatul de afișare a valorilor conținute în câmp. Șablonul poate conține anumite caractere, în funcție de tipul câmpului.

Caracterele „!”, „X” se pot folosi la afișarea textelor. De exemplu, pentru fereastra din figura anterioară se cere ca indiferent de caracterele tastate șirul cu specialitatea persoanei să fie convertit la majuscule.

- caption permite introducerea numelui lung pentru câmp, folosit în ecranele Browse sau în obiectele care îl conțin³ (CAPTION);
- regula de validare la nivel de atribut și mesajul asociat (RULE, MESSAGE); valoarea implicită a câmpului (DEFAULT);
- comentariile care se vor afișa pe lina de stare (COMMENT) interesează doar proiectantul, în vederea unei revizuirii a structurii;

³ În comanda Browse pentru asocierea unui alt identificator pentru o coloană decât numele câmpului se folosește opțiunea H.

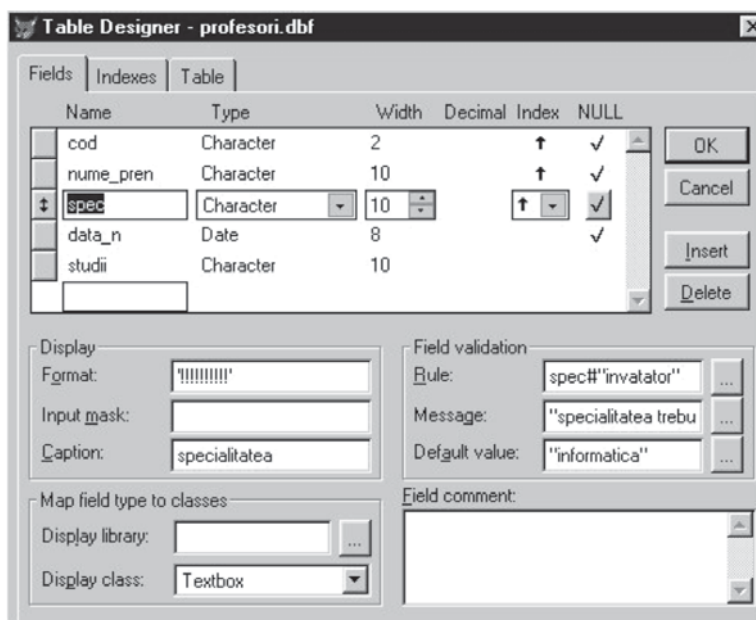


Figura 13-5: Tab-ul Fields din Table Designer

g) obiectul⁴ și clasa de obiecte corespondente, astfel încât la compunerea machetei de introducere sau de vizualizare să fie automat definit obiectul vizual respectiv când se folosește câmpul. În mod implicit, câmpurile au asociate obiecte de tip TextBox, iar câmpurile Memo – obiecte EditText.

■ **Tab-ul Indexes** permite fixarea indecșilor asociați tablei.

Așa cum știm deja, un index permite accesul ordonat la tabelă. Indecșii asociați unei table sunt precizați prin nume (NAME), sensul ordonării, ascendent sau descendent (ORDER), tip (TYPE), expresia de indexare (Expression), precum și condiția de filtrare (FILTER).

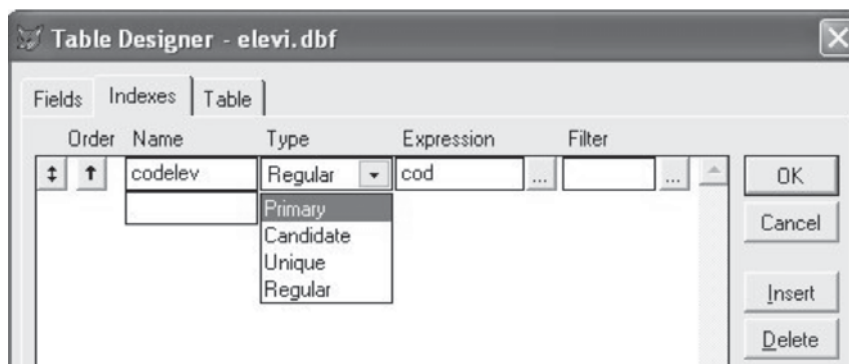


Figura 13-6: Tab-ul Indexes

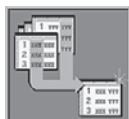
⁴ Asocierea obiectelor este referită în lecțiile de proiectare a formularelor.

Reamintim tipurile de indecși care pot fi folosiți într-o tabelă inclusă într-o bază de date:

- **Index primar (Primary index)** – asigură introducerea valorilor unice într-o tabelă. O tabelă are un singur index primar;
- **Index candidat (Candidate index)** – tabela are deja un index primar, dar dorim verificarea valorilor unice și în alt câmp;
- **Index obișnuit (Regular index)** – permit înregistrări duplicate;
- **Index unic (Unique indexes)** – selectează ordinea de parcurgere bazată pe prima apariție a valorii în câmpul specificat.

■ **Tab-ul Table** permite specificarea condițiilor de validare la nivelul întregului articol (Rule) și al mesajului asociat situației de eroare (Message), a declanșatorului (*trigger*) pentru inserarea articolelor (Insert trigger), pentru corecții (Update trigger) și pentru ștergere (Delete trigger). Despre declanșatoare vom discuta într-o temă specială.

b. Proiectarea rapidă a tabelor cu Table Wizard



Pasul 1: Alegeți una dintre tabelele predefinite; fixați câmpurile care vor forma structura.



Pasul 2a: Asociați tabela unei baze de date; baza de date este deschisă implicit. Generați tabela ca izolată (Free Table). Dați tablei un (alt) nume.

Pasul 2b: Vizualizați structura tablei și eventual modificați-o prin selectarea anumitor câmpuri, schimbarea numelui, a tipului, a lungimii etc.

Pasul 3: Precizați indecșii și câmpul folosit drept cheie primară. În exemplu, câmpul Cod este index primar.

Pasul 4: Puteți izola tabela curentă sau o puteți lega de alte tabele din bază. Se fixează relațiile între tabela curentă și unele dintre tabelele bazei de date active. Există mai multe modalități de fixare a relației. În exemplul nostru am fixat relația între cititori și cărți de tip 1-n, în sensul că un cititor poate împrumuta mai multe cărți.



Pasul 5: Salvați definițiile tablei și închideți utilitarul.

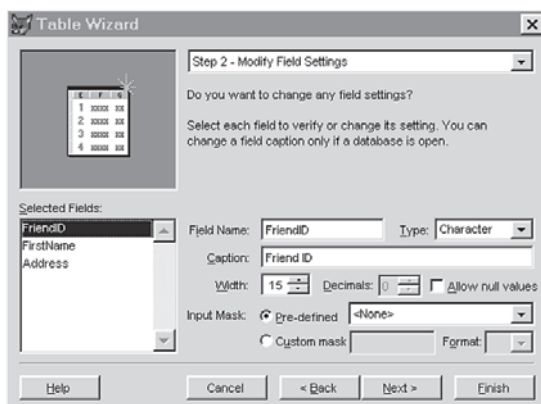


Figura 13-7: Pasul 2 din Table Designer

c. Meniul contextual asociat tabelii

Tabelele vizualizate în fereastra Database Designer au asociat un meniu accesibil prin clic dreapta și care permite:

- editarea în fereastra Browse a tabelii,
- ștergerea tabelii din baza de date sau chiar de pe disc (**DELETE**),
- micșorarea tabelii în fereastră (**COLAPSE**),
- modificarea structurii tabelii (**MODIFY**), apelul utilitarului **Help**.

Exemplu

Dorim crearea unei baze de date cu două tabele, alese din cele oferite de produsul FoxPRO: Books și Friends.

Vom parcurge următorii pași:

1. Vom deschide utilitarul Database Designer selectând **File, New, Database**. Fixăm numele bazei de date Biblio.
2. Pentru includerea fiecăreia dintre cele două tabele vom apela Table Wizard din fereastra de dialog pentru crearea unei tabelii noi: **Database, New Table, Wizard**.
3. Vom parcurge pașii necesari alegerii tabelii, selectării câmpurilor, precizării destinației.
4. Observăm în fereastra Database Designer cele două tabele.
5. Pentru ca tabela Books să fie prezentă numai cu numele se apelează din meniul contextual opțiunea **collapse**.

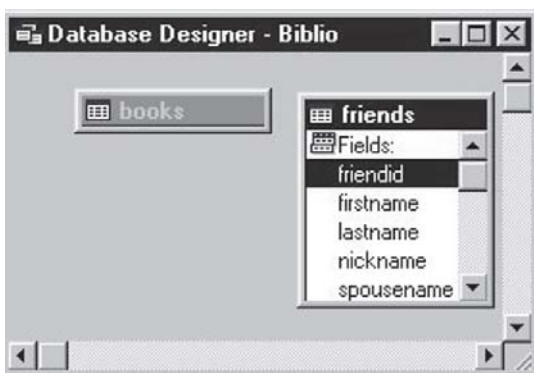


Figura 13-8: Crearea unei baze de date cu două tabele



sarcini de laborator

1. Folosind *fereastra de comenzi* creați o bază de date FOTBAL cu tabelele MECIURI, ECHIPE, JUCATORI. Stabiliți câmpurile necesare evidenței tuturor meciurilor dintr-un campionat, a jucătorilor și echipelor de fotbal.
2. Folosind Database Designer și Table Designer proiectați o bază de date TURISM cu tabelele UNITATI (care va reține toate unitățile de cazare cu numele, categorie, adresa, dacă are sau nu piscină, restaurant, etc), LOCURI (care va reține toate locurile/camere din unitățile de cazare cu numărul de paturi, dacă au sau nu televizor, telefon, orientare, pret, etc.) Stabiliți proprietăți pentru câmpuri, valori implicite, cheia fiecărei tabelii.
3. Deschideți ambele baze de date și treceți controlul de la una la alta. Faceți o modificare în structura unei tabelii. Inchideți odată ambele baze de date.

Fixarea relațiilor persistente între tabelele unei baze de date

Relațiile între tabelele dintr-o bază de date sunt **permanente**⁵ (**persistente**), în sensul că intră automat în vigoare la deschiderea bazei de date și sunt valabile pe tot parcursul lucrului cu baza de date. Relațiile sunt memorate ca părți integrante ale bazei de date.

Stabilirea unei relații presupune:

a) *identificarea tipului relației*: 1-1 (unu-la-unu) sau 1-n (unu-la-n)

Atenție: Tipul relației va fi dat de tipul indexului din tabela copil. Dacă indexul este cheie unică (de tip *unique*, *primary*, *candidate*), atunci relația va fi 1-1; dacă este cheie comună (de tip *regular*), atunci relația va fi 1-n.

b) *pregătirea tabelor*: tabela părinte se indexează unic după cheia de legătură (câmpul index va fi de tip *primary* sau *candidate*); tabela copil se indexează unic (pentru o legătură 1-1) sau normal (pentru o legătură 1-n) după cheia de legătură. Ambii indecși trebuie să fie de același tip și să aibă aceleași expresii de indexare.

c) *proiectarea vizuală a legăturii* se face prin fereastra Database Designer sau prin caseta de dialog Data Session. Astfel:

- în fereastra *Database Designer* ne poziționăm în tabela părinte, pe indexul primar și executăm tragere și plasare către tabela copil peste indexul folosit drept cheie de legătură.
- deschidem fereastra *Data Session* din meniul principal Window, apăsăm butonul **Relations** (pentru stabilirea legăturii) și introducem expresia de legătură dintre cele două tabele.

Exemplu

Folosirea ferestrei Database Designer la proiectarea legăturilor

Deschidem baza de date Scoala și, în fereastra Database Designer, observăm tabelele incluse. Să presupunem că sunt cele trei din figura 13-9.

Vom fixa legăturile Profesori→1,1→clase pentru diriginte și Clase→1,n→elevi.

Pasul 1. Ne poziționăm pe tabela Profesori și verificăm sau adăugăm indexul Codprof de tip *primary* prin fereastra Table Designer.

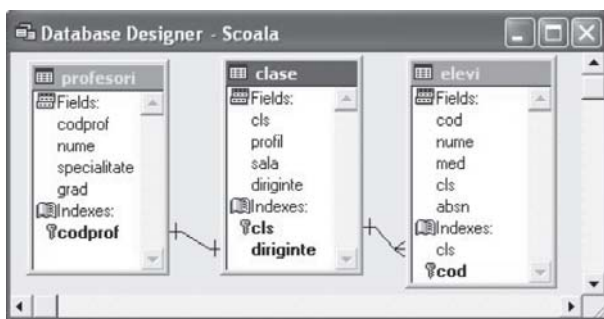


Figura 13-9: Tabelele incluse în baza de date

⁵ Reamintim că relațiile realizate prin comanda **SET RELATION** sunt temporare, disparând la terminarea programului sau a sesiunii de lucru.

Pasul 2. Ne poziționăm pe tabela Clase și folosim câmpul diriginte drept index *candidate* pentru că în tabela clase cheia diriginte este unică. De asemenea, indexăm (sau verificăm dacă există) indexul *primary pe* câmpul cls.

Pasul 3. Ne poziționăm pe tabela Elevi și indexăm după câmpul Cod (cheia primară) și după câmpul cls index de tip *regular*.

Pasul 4. Fixăm relația Profesori→1,1→Clase prin tragere și plasare de la indexul profesori.codprof către cheia clase.diriginte.

Pasul 5. Fixăm relația Clase→1,n→Elevi prin manevra tragere și plasare de la indexul clase.cls către indexul elevi.cls.

Observăm în fereastra Database Designer legătura proiectată.

Folosirea ferestrei Data Session

1. Deschidem fereastra Data Session din meniul **Window**.
2. Ne poziționăm pe tabela Clase și apăsăm butonul **Relation**.
3. Selectăm tabela copil (Elevi).
4. Introducem expresia de legătură: Clase.cls=elevi.cls.

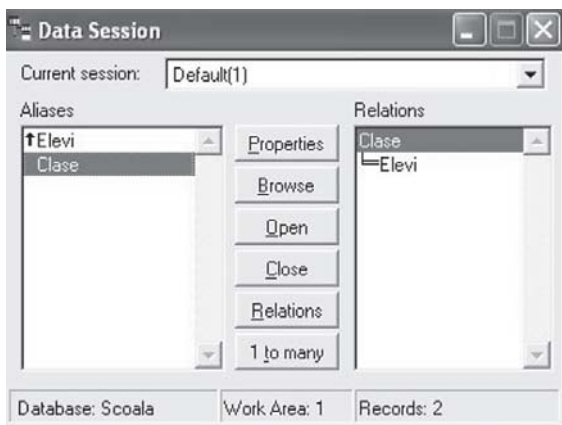


Figura 13-10: Fereastra Data Session

Ștergerea sau modificarea unei relații

Pentru editarea unei relații se poate folosi meniul contextual deschis atunci când, poziționați fiind pe legătură (ea este mai intens colorată), executați clic dreapta. Comanda **Edit** deschide un ecran de dialog.

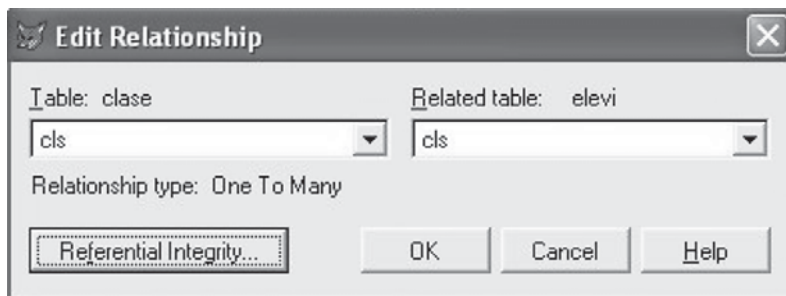


Figura 13-11: Fereastra Edit Relationship

Ștergerea unei relații se poate face și prin apăsarea tastei Delete.



sarcini de laborator

I. La Liceul de Informatică se organizează preselecția pentru „MISS BOBOC“.

Participanții și rezultatele la cele 3 probe artistice sunt reținute într-o tabelă cu numele Miss.dbf, iar membrii juriului sunt înscriși în tabela Juriu (cod, nume, ocupația, vârsta, sex). Avem în vedere următoarele:

- Numele candidatei trebuie să fie trecut obligatoriu cu majuscule.
- Proba 1 se notează cu note între 5 și 10 cu două zecimale.
- Proba 2 se va nota cu calificative {ffb, fb, b, suf, insuf}.
- Proba 3 se notează cu puncte între 1 și 100.
- O persoană este admisă dacă are la prima probă cel puțin 5.00, a obținut la proba a doua cel puțin suficient, iar la a treia mai mult de 60 puncte.
- Se vor reține și anumite informații despre candidată (înălțimea, greutatea, culoarea ochilor, lungimea părului, fotografia, mesajul verbal adresat comisiei etc.)
- Ocupația unui membru al comisiei nu trebuie să fie „elev“ sau „student“.
- Vârsta membrilor comisiei trebuie să fie cuprinsă între 25-30 ani.
- Codificarea pentru câmpul Sex se va face prin „F“ sau „B“.
- Codul membrilor juriului va avea pe prima poziție o literă mare, apoi 2 cifre și iar o literă.

1. Proiectați baza de date, incluzând tabelele necesare.
2. Populați-o cu date semnificative.

II. O bază de date este formată din următoarele fișiere:

PROFESORI (cod_prof, nume, specialitate, grad_didactic);

STUDENTI (cod_stud, nume, facultate,an);

CURSURI (cod_curs, nume, nr_ore, cod_prof);

ALEG (cod_stud, cod_curs) – opțiunile studenților.

Se cere:

1. Afișați numele cursurilor predate de fiecare profesor.
2. Afișați lista studenților la fiecare curs.
3. Verificați corectitudinea secvenței următoare! Ce își propune să obțină?

```
sele aleg
index on cod_curs tag cc unique
sele curs
set relationoff into ale g
sele aleg
set relation to cod_curs into curs
list curs.nume
```

4. Realizați structura următoare prin comenzi adecvate și dați o interogare care să necesite aceste relații.



5. Pentru toti studentii care și-au ales deja cursuri, aflați numele lor și al cursurilor alese.
6. Afișați cursurile fără studenți.
7. Afișați numai profesorii la cursurile cărora s-au înscris studenți. Împreună cu numele cursului se pot afișa și numele studenților care au optat pentru curs?

Proceduri stocate

Procedurile stocate sunt memorate în baza de date și sunt deschise odată cu aceasta. Operația de adăugare a procedurii la baza de date este precedată de scrierea ei într-un fișier text.

Procedură	Efect
MODIFY PROCEDURE	Editarea unei proceduri.
APPEND PROCEDURE FROM<fis.txt> [OVERWRITE]	Adăugarea unei proceduri stocate la baza de date curentă dintr-un fișier <Fis.txt>, cu sau fără suprascriere.
DISPLAY PROCEDURES	Afișarea procedurilor memorate din baza curentă.
COPY PROCEDURE TO <fis.txt> [ADDITIVE]	Copierea procedurii stocate într-un fișier text, de unde poate fi manipulată.

Pentru a introduce o funcție utilizator drept regulă de validare, procedura stocată trebuie să existe. În caz contrar, FoxPro va respinge referința la funcția utilizator.

Orice apel al funcției de validare trebuie să returneze un rezultat logic. Dacă în urma validării câmpurilor este returnată valoarea .F., FoxPro păstrează indicatorul de înregistrări în aceeași linie și nu salvează modificarea făcută asupra ei.

Pentru validare se poate folosi funcția **GETFLDSTATE ()** care determină dacă un câmp al tabelii sau view-ului s-a modificat.

Puteți folosi declanșatorii pentru calcule sau validări suplimentare, puteți trimite e-mail-uri unui compartiment de aprovizionare atunci când stocul dintr-un articol atinge o anumită valoare redefinită, sau puteți crea jurnale cu toate modificările efectuate asupra unui tabel.

Exemple

Open database D1	crează baza de date, care este deschisă automat
Create table t1 free (cmp M)	tabela temporară
Append blank	adaugă o linie vidă
Replace cmp with "procedure; fictiv"+chr(13)+chr(10)	pune în coloana Memo linia de definiție a unei proceduri fictive
Copy memo cmp to fis.txt	copiază conținutul câmpului Memo într-un fișier
Use	închide tabela temporară t1
Append procedure from; fis.txt	adaugă la bază fis.txt ca fiind procedură stocată
Display procedures	afișează procedurile bazei D1

Delete file t1.dbf	șterge de pe disc manevrele
Delete file t1.fpt	
Modify procedure	apelează editorul pentru modificarea procedurilor
Close database	închide baza de date

Alt exemplu: generarea automată a codului la adăugarea unui nou articol.

În tabela Elevi dorim generarea automată a codului la adăugarea unui nou elev. Practic, noul cod va fi ultimul cod +1.

Pasul 1. Vom deschide fereastra de editare a procedurilor stocate cu **MODIFY PROCEDURE** și vom scrie procedura următoare:

```
procedure pune_cod
calculate max(elev.cod) to codmax
if empty(codmax)
=messagebox("nu sunt coduri")
return 1
else
=messagebox("codul ultim="+str(codmax))
return codmax+1
endif
endproc
```

Observație:

Fereastra de editare a procedurilor stocate într-o bază de date se poate deschide și direct, selectând Database, Edit Stored Procedures.

Pasul 2. Vom apela procedura ca valoare implicită a câmpului în momentul unei adăugări; de exemplu, codul persoanei va fi automat generat.

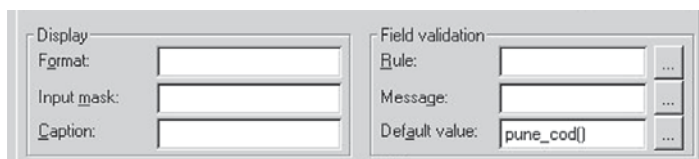


Figura 13-12: Generarea automată a codului

Declanșatoare

Declanșatoarele (triggers) sunt acele proceduri stocate, rulate după o operație de inserare, actualizare sau ștergere, care atașează anumite acțiuni anumitor evenimente. Se folosesc în special la păstrarea integrității referențiale într-o bază de date.

CREATE TRIGGER ON <dbf> FOR DELETE/INSERT/UPDATE AS <expl>	Crearea unui declanșator atașat tablei <dbf> din baza de date curentă. Dacă <expl> este .T., atunci se dă curs operației de inserare, actualizare, ștergere.
---	---

DELETE TRIGGER ON <dbf> FOR DELETE/ INSERT/UPDATE	Ștergerea unui declanșator asociat tablei <dbf> incluse în baza curentă.
--	---

Pentru fiecare tabelă a unei baze de date se pot crea maxim 3 declanșatoare. Procedura de validare indicată prin <expl> va trebui să returneze .T. sau .F. Expresia poate fi și o funcție utilizator sau o procedură memorată (creată cu **MODIFY PROCEDURE**).

Atenție, baza de date curentă trebuie să fie deschisă în mod exclusiv!

Exemplu

Pentru tabela ELEVI dorim ca verificarea mediei la actualizare să nu fie mai mare ca 10:

Open database x	&& elevi.dbf(ume,clasa,media)
Use elevi	
Create trigger on elevi for update AS media<=10	&&Creare declanșator
On error	&& restaurează gestionarul de erori
Replace all media with 11.88	&& se afișează mesaj de eroare
Replace all media with 4.67	&& se acceptă valoarea !

Proiectarea regulilor de integritate a bazei de date

Bazele de date permit, așa cum știm din lecțiile anterioare, verificarea unor condiții la orice acțiune de actualizare a datelor, astfel încât să se asigure corectitudinea datelor – validări la nivelul fiecărui câmp sau la nivelul unei înregistrări. Se pot fixa și anumite condiții pentru tabelele legate, astfel încât să se asigure integritatea bazei de date.

Premiza de bază este că valorii unei chei externe dintr-o tabelă copil trebuie să-i corespundă o cheie de căutare sau o cheie primară în tabela părinte. Mecanismul de integritate referențială tratează înregistrările care nu îndeplinesc aceste condiții drept invalide. Prin analogie, puteți fi un părinte fără fii, dar nu puteți avea un fiu fără a fi părinte.

Apelul utilitarului **Referential Integrity Builder** se face din meniul contextual asociat relației sau selectând **Database, Edit Referential Integrity**. La apelul utilitarului se deschide o fereastră de dialog cu 3 tab-uri care permit editarea condițiilor pentru actualizarea datelor, ștergerea și inserarea.

În tab-ul **Rules for Inserting** se pot stabili regulile pentru adăugarea articolelor în tabela copil. Opțiunile disponibile sunt:

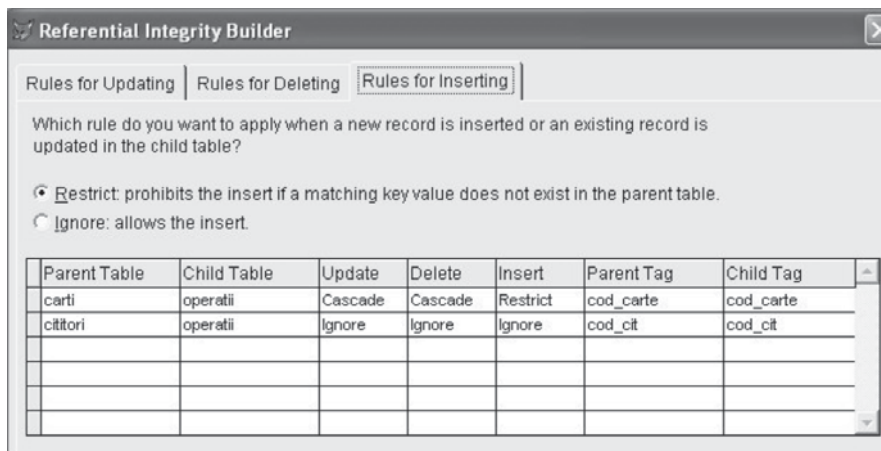


Figura 13-13: Tab-ul Rules for Inserting

- **Restrict** – interzice introducerea articolelor în tabela copil dacă în tabela părinte nu există nici o înregistrare cu aceeași valoare a cheii.
- **Ignore** – permite inserarea articolelor fără verificarea valorii cheii străine în tabela părinte.

În tab-ul **Rules for Deleting** se stabilesc reguli atunci când se ștege un articol în tabela părinte. Opțiunile disponibile sunt:

- **Cascade** – șterge toate articolele cu aceeași valoare ca a cheii străine;
- **Restrict** – interzice ștergerea dacă în tabela copil sunt articole cu aceeași valoare a cheii străine ca și cea a cheii șterse;
- **Ignore** – permite ștergerea din tabela părinte și lasă articolele „orfane“ în tabela copil.

Atenție! Întotdeauna trebuie bine gândită acțiunea de ștergere. Desigur, dacă ștergem o carte din evidențele noastre, toate operațiile asupra ei ar trebui șterse. Dar acestea sunt operații trecute, reținute poate în fișierul Operații pentru a se vedea activitatea, succesul unei cărți. Faptul că ea este pierdută nu-i scade din valoare! Pe de altă parte, putem proiecta mecanismul de protecție astfel ca să nu putem șterge o carte dacă ea este împrumutată cuiva!

Tab-ul **Rules for updating** permite alegerea posibilităților de actualizare a articolelor atunci când se modifică cheia primară a unui articol în tabela părinte. Opțiunile disponibile sunt:

- **Cascade** – actualizează toate articolele legate în tabela copil cu noile valori ale cheii modificate.
- **Restrict** – interzice modificarea cheii dacă sunt articole legate în tabela copil.
- **Ignore** – actualizează cheia fără verificarea corespondenței din tabela copil.

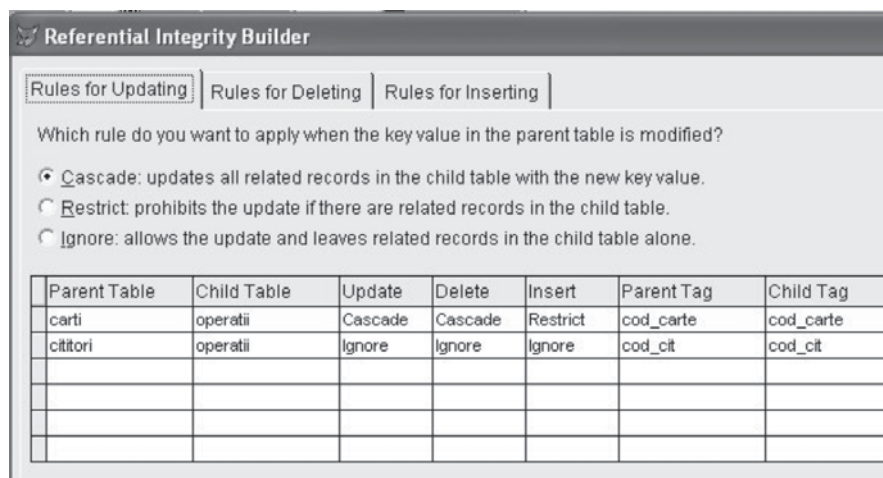


Figura 13-14: Tab-ul Rules for Updating

Dacă modificăm structura tabelelor implicate în mecanismul de integritate referențială, indecșii sau relațiile permanente, trebuie să rulăm din nou utilitarul Referential Integrity Builder. În acest fel, codul sursă din baza de date va fi revizuit.

Generarea unui cod pentru fiecare tip de declanșator se poate vizualiza prin deschiderea ferestrei de proceduri stocate (selectând **Database, Edit Storage Procedure**).

Dorim să fixăm restricțiile de integritate. Astfel:

a. Vom interzice adăugarea în tabela Operatii a unei înregistrări pentru o carte, să zicem cu valoarea inexistentă în tabela Carti. Acest lucru impune folosirea unui declanșator pentru inserare de tip „Restrict”.

b. Modificarea codului unei cărți în tabela Carti va determina modificarea peste tot a cheii cod_carte la noua valoare. Acest lucru impune folosirea unui declanșator pentru actualizare de tip Cascade.

Ștergerea unei cărți din fișierul Carti va determina ștergerea tuturor articolelor din fișierul Operatii. Acest lucru impune folosirea unui declanșator pentru ștergere de tip Cascade.

Pași:

1. Fixăm relația Carti→1-n→Operatii prin câmpul cod-carte.
2. Deschidem Edit Referential Integrity și fixăm pentru inserare declanșatoare de tip Restrict, iar pentru actualizare și ștergere folosim tipul Cascade.
3. **Verificăm** funcționarea declanșatoarelor astfel: deschidem în fereastra Browse tabelele Carti și Operații. Marcăm pentru ștergere o linie cu un cod-carte care apare în tabela Operații. Articolele referite de cod sunt automat marcate!

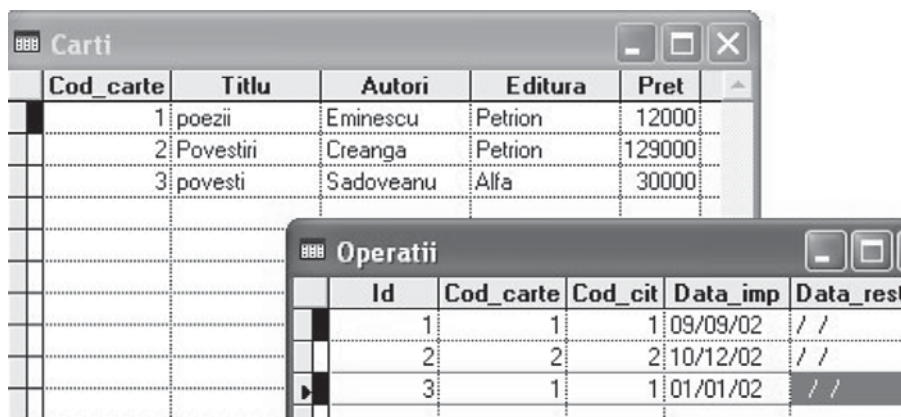


Figura 13-15: Verificarea funcționării declanșatoarelor

4. Observăm plasarea declanșatoarelor automat în cele două tabele.

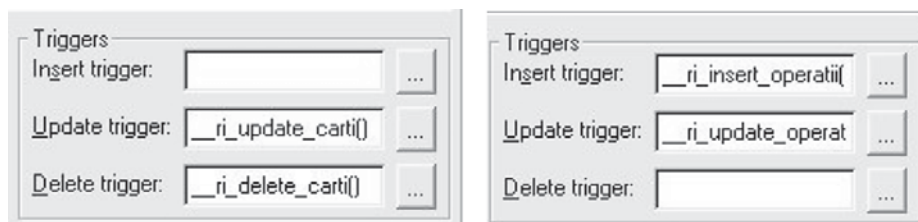


Figura 13-16: Plasarea automată a declanșatoarelor în cele două tabele

- c. Deschidem fereastra Table Designer, tab-ul Table.

Observație. Pentru a putea fixa restricțiile de integritate este necesar ca imediat după crearea unei baze de date (prin **File, New, Database**) ocazie când se deschide fereastra Database Designer, să dăm comanda **Database, Clean up database** și apoi să adăugăm sau să proiectăm tabelele de date.



sarcini de laborator

I.

1. Folosind fereastra de comenzi, creați baza de date Școală cu tabelele: Elevi (clasă, nume, adresă), Clase (clasă, sală, diriginte), Săli (sală, inventar, poziție) Profesori (nume, clasă, specialitate).
2. Stabiliți relațiile Elevi-Clase, Săli-Clase, Profesori-Clase, fixând tipul relației și cheia de legătură.
3. Deschideți baza de date în fereastra Database Designer și vedeți legăturile create.
4. În fereastra de comenzi, scrieți comenzile pentru:
 - a) scoaterea fișierului Săli din contextul bazei de date;
 - b) modificarea structurii fișierului Clase;
 - c) ștergerea legăturii Elevi-Clase.

II.

În vederea realizării orarului, sunt proiectate tabelele următoare: Orar (ziua, ora, codul disciplinei, codul profesorului, codul clasei), Profesori (codul profesorului, numele, catedra), Discipline (codul disciplinei, numele, catedra, nivelul sau anul/anii la care se predă disciplina respectivă).

Discipline

bi	Bazele informaticii	Info	9,10,11,12
la	Informatică aplicată	Info	10,11,12
so	Sisteme de operare	Info	10

Profesori

P1	Popescu Emilian	Mate
P2	Zaharescu Vasilache	Info

Orar

L	8	bi	P2	9a
L	9	so	P2	10b

Convenții:

- Ora poate fi în intervalul 8-14.
- Ziua se codifică pe două caractere și aparține mulțimii {L, Ma, Mi, J, V}.
- Într-o catedră sunt mai multe discipline.
- Același profesor poate să aibă ore la diferite clase, dar nu în același timp. El poate să predea orice disciplină din catedra sa.

Sarcini:

1. Creați baza de date cu numele ORE.
2. Adăugați tabelele menționate anterior. Fixați cheile unice ale fiecăreia.

3. Stabiliți relațiile necesare prin fereastra de comenzi.
4. Încărcați cu date fișierele Profesori și Discipline.
5. Pentru baza de date ORE fixați legăturile permanente, știind că:
 - a) Un profesor poate să predea mai multe discipline.
 - b) O disciplină este regăsită în orar la mai multe clase, dar la ore diferite.
 - c) O disciplină poate fi predată de profesori diferiți, dar în cadrul aceleiași catedre.
6. Scrieți comenzile necesare obținerii situației încadrării tuturor profesorilor care predau disciplina X (cine, la ce clasă, când).
7. Scrieți comenzile pentru situația încadrării unui profesor X (ce discipline predă, când, la ce clase).
8. Scrieți comenzile prin care să aflați dacă există suprapuneri în orar pentru anumite discipline sau profesori.
9. Aflați încadrarea tuturor claselor a XII-a (profesorii și numele disciplinei predate).
10. Proiectați regulile de integritate pentru baza de date ORE astfel încât:
 - a) La adăugarea unei linii în orar:
 - codul profesorului să se găsească în fișierul Profesor;
 - codul disciplinei să se găsească în fișierul Discipline;
 - anul de studiu din codul clasei să corespundă unui nivel de predare a disciplinei respective;
 - disciplina predată de un profesor să corespundă catedrei acestuia.
 - b) La ștergerea unui profesor să dispară toate orele sale.
 - c) La ștergerea unei discipline să dispară toate orele din orar cu disciplina respectivă.
 - d) La ștergerea unei catedre să fie șterși toți profesorii care au încadrare pe catedra respectivă, precum și toate disciplinele asociate catedrei. În orar vor fi șterse orele corespunzătoare profesorilor și disciplinelor.
 - e) La modificarea codului unei discipline se va modifica peste tot acest cod în orar.
 - f) Modificarea codului unui profesor determină modificarea peste tot a codului în orar.
 - g) Adăugarea unei înregistrări în Profesori nu afectează celelalte fișiere.
 - h) Adăugarea unei înregistrări în Discipline nu afectează celelalte fișiere.



sarcini pentru realizarea unui mini-proiect

Proiectați sau reprojectați tabelele bazei de date specifice fiecărui proiect, având în vedere posibilitatea fixării condițiilor de validare, a valorilor inițiale etc.

Fixați relațiile persistente dintre tabelele bazei de date.

Fixați regulile de integritate referențială pentru baza de date.

Interogarea bazelor de date

- Definirea interogărilor prin comanda **SELECT**
- Proiectarea vizuală a interogărilor
- Proiectarea fișierelor View
- Query Designer

O **interogare (cerere)** este o solicitare directă de date, fără indicarea modului de obținere.

O cerere SQL se poate lansa din fereastra de comenzi sau din interiorul unui program FoxPro, pentru că funcționează ca orice altă comandă, dar se poate proiecta în mod interactiv cu ajutorul utilitarului Query Designer.

Definirea interogărilor prin comanda **SELECT**

Comanda **SELECT** permite specificarea datelor care vor constitui ieșirea din interogare, precum și sursa acestora. **Cum** se vor obține aceste date este sarcina optimizatorului de cereri Rushmore.

```
SELECT [ALL/DISTINCT]
[<col1>[AS<nume>], <col2> [AS <nume2>..]..]
FROM <lista-fis>
    [INTO <destinatie>/ TO FILE <fis.dbf>]
[WHERE <conditie>]
[GROUP BY <lista_chei>[ HAVING <cond>]]
[ORDER [BY] <exp>]
```

- Clauza **ALL/DISTINCT** determină prelucrarea tuturor înregistrărilor (**ALL**) sau numai a articolelor unice (**DISTINCT**).
- Clauza **<col1> [AS <nume1>]...** permite definirea coloanelor care vor constitui ieșirea din interogare. Coloanele pot fi câmpuri aparținând tabelor definite în clauza **FROM**, constante, expresii, funcții utilizator. Coloanele pot primi un alt nume prin clauza **AS**.
Pot fi utilizate funcții cum sunt: **AVG(<exp>)** care calculează media aritmetică, **COUNT(<art selectat>)** care numără selecțiile, **SUM(<art-selectat>)** care calculează suma, **MIN(<art-selectat>)**, **MAX(<art-selectat>)** care determină extremul.
- Clauza **FROM** specifică lista fișierelor de intrare în prelucrare.

Exemplu: Fie Baza de date Școala cu tabelele ELEV1, CLASE:

Cerere	Rezolvare
1. Numele tuturor elevilor	<code>select nume from elevi</code>
2. Numele claselor (distincte!) din școală	<code>select distinct cls as cod clasa from elevi</code>
3. Tot conținutul fișierului ELEV1	<code>select all from elevi select * from elevi</code>

4. Media generală a elevilor	<code>select avg(med) as med_gen from elevi</code>
5. Numărul elevilor din școală	<code>select count(*) as nr_elelevi from elevi</code>
6. Numărul claselor din școală	<code>select count(distinct cls) from elevi</code>

- Destinația rezultatelor este specificată prin două clauze: **INTO/TO**, dintre care **INTO** este prioritară. Clauza **INTO <dest_into>** determină forma de stocare a datelor; lipsa clauzei permite afișarea într-o fereastră **BROWSE** a datelor. **<Dest_into>** poate fi: **ARRAY <tablou>/CURSOR< fis>/DBF <fis.dbf>**. Forma de stocare cursor este o tabelă temporară, de tip Read Only, ștersă automat în momentul închiderii ei. Clauza **TO** este folosită când lipsește clauza **INTO**. **<Dest_to>** poate fi: **TO FILE <fis.txt> [ADDITIVE] / TO PRINTER [PROMPT] / TO SCREEN**, unde **TO FILE** direcționează ieșirea către un fișier ASCII (fie prin suprascriere, implicit, fie prin adăugarea datelor la vechiul conținut, folosind clauza **ADDITIVE**), **TO PRINT** către imprimantă, iar **TO SCREEN** către ecran.

Exemple:

Cerere	Rezolvare
Afișați elevii claselor la imprimantă	<code>select cls, nume from elevi to print</code>
Copiați numai numele și codul elevilor într-o altă tabelă	<code>select cod, nume from elevi; into dbf manelevi.dbf</code>

- Clauza **WHERE <cond>** permite introducerea legăturilor între tabele și a filtrelor. Condițiile **<cond>** cer sistemului FoxPro să includă anumite înregistrări în rezultatele cererii.

Exemple:

Cerere	Rezolvare
Care sunt elevii cu medii între 8 și 10 din clasa „12A“?	<code>select nume, med from elevi; where med between 8 and 10 and cls="12A"</code>
Care sunt elevii cu media 10 precum și numele diriginților lor?	<code>select elevi.nume, elevi.cls, clase.diriginte; from elevi, clase where elevi.cls=clase.cls; and elevi.med=10</code>
Care sunt elevii dirigințelui X?	<code>Accept 'nume profesor diriginte?' to x Select elevi.nume, elevi.med from elevi, clase; where clase.cls=elevi.cls and clase.diriginte=x</code>

- Clauza **GROUP** permite gruparea rezultatelor după lista de câmpuri.

Exemplu:

Cerere	Rezolvare
Care sunt elevii cu media 10 pentru fiecare clasă (și numele diriginților lor)?	<code>select elevi.cls, clase.diriginte; elevi.nume from elevi, clase; where elevi.cls=clase.cls and; elevi.med=10 group by cls</code>

- Clauza **HAVING** permite introducerea unor restricții de afișare a grupului.

Exemplu:

Cerere	Rezolvare
Care este numărul elevilor pe clase, numai de la profilul „info“?	<pre>select elevi.cls, count(*) from; elevi, clase where elevi.cls=clase.cls; group by cls having clase.profil="info"</pre>

- Clauza **ORDER BY <exp-ordo> ASC/DESC** permite specificarea expresiei de ordonare și a sensului ordonării.

Exemplu:

Cerere	Rezolvare
Lista alfabetică a elevilor dirigintei „Popa“.	<pre>select elevi.ume from elevi, clase where; elevi.cls=clase.cls and clase.diriginte="Popa"; order by elevi.ume asc</pre>



sarcini de laborator

Considerăm o firmă care produce și comercializează produse din pielărie.

1. Proiectați o bază de date necesară pentru compartimentul comercial în activitatea de aprovizionare.

Se cere:

- a. evidența furnizorilor de materiale (care sunt, ce materiale oferă, la ce prețuri, condiții de transport, alte facilități);
- b. evidența contractelor de aprovizionare (din momentul încheierii lor se înregistrează ca obligații de livrare de către furnizor a unor materiale în cantitățile și la termenele fixate);
- c. evidența facturilor. Odată cu livrarea materialelor, furnizorul întocmește o factură în care se înscrie contractul în baza căruia se face livrarea, materialele trimise, prețuri, condiții de plată.
- d. evidența achitărilor. Pentru livrările făcute, firma trebuie să plătească. Plata facturii se înregistrează pe un document numit chitanță. Chitanțele identifică factura, data achitării și suma. O factură poate fi achitată prin mai multe chitanțe. Desigur, putem achita și prin bancă!

2. Calculați necesarul de aprovizionat pe baza planului de fabricație și a consumului unitar de materiale. Se va ține cont de stocurile existente!

3. Aflați prin interogări **SELECT**:

- a. Care sunt furnizorii pentru materialul x?
- b. Care este cantitatea totală contractată pentru materialul x?
- c. Care dintre furnizorii pentru materialul x oferă prețul cel mai mic?
- d. Câte facturi au fost trimise pentru contractul x?

- e. Care este valoarea totală a contractelor încheiate cu furnizorul X?
- f. Există contracte neonorate la termen sau cantitate?
- g. Care sunt furnizorii care nu au facturi?
- h. Există facturi neachitate?
- i. Care este situația centralizată a achitărilor pentru furnizorul X: total valoric facturi, total sume achitate, diferența?
- j. Care sunt furnizorii care nu au nici un contract?
- k. Există contracte încheiate cu furnizori neînregistrați în baza de date?
- l. Există materiale necesare, pentru care nu există furnizori sau contracte?

Exemplu

Sugestie de rezolvare a punctului 2:

Necesar de aprovizionat¹

Necesarul de aprovizionat se determină în funcție de stocul existent și necesarul de materiale pentru realizarea planului de producție. Tabelele necesare sunt Plan (cod-produs, cantitate-planificată), Stoc (cod material, stoc-existent) și Consum-unitar (cod-produs, cod-material, cantitatea de material consumată pentru realizarea unității de produs).

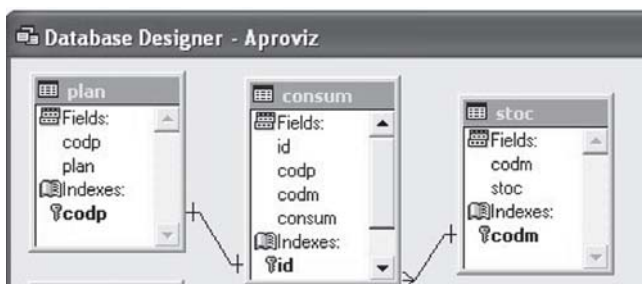


Figura 14-1: Proiectarea bazei de date

Pasul 1. Proiectăm baza de date (pe care o numim Aproviz) cu tabelele Plan, Consum, Stoc. Fixăm relațiile plan→1,n→consum; stoc→1,n→consum.

Pasul 2. Calculul necesarului de materiale pentru realizarea planului se va face înmulțind consumul unitar de material cu planul de producție. Relația dintre cele două tabele permite asocierea cantității planificate pentru fiecare consum în parte; baza de date curentă este Aproviz.

```
SELECT Plan.plan, Consum.codp, Consum.codm, Consum.consum, ;
  Plan.plan*Consum.consum AS necplan;
FROM aproviz!plan INNER JOIN aproviz!consum ;
ON Plan.codp = Consum.codp;
ORDER BY Consum.codm;
INTO DBF fnecplan
USE
ADD TABLE fnecplan && trecem tabela nou creată în baza de date
```

Pasul 3. Calculăm necesarul total de materiale pentru fiecare material.

```
SELECT fNecplan.codm, SUM(fNecplan.necplan) AS nectotal;
FROM aproviz!fnecplan;
GROUP BY fNecplan.codm;
INTO DBF fnectotal
USE
ADD TABLE fnectotal
```

¹ Urmăriți alte variante de rezolvare la sfârșitul manualului.

Pasul 4. Scădem din stocul existent necesarul total și aflăm necesarul de aprovizionat.

```
SELECT Stoc.codm, Stoc.stoc-fnecstotal.necstotal;  
FROM aproviz!fnecstotal INNER JOIN aproviz!stoc;  
ON fnecstotal.codm = Stoc.codm;  
GROUP BY fnecstotal.codm;  
ORDER BY Stoc.codm;  
INTO DBF fnecaprov  
USE  
ADD TABLE fnecaprov && trecem fișierul fnecaprov în baza de date
```

Proiectarea vizuală a interogărilor

O interogare este o modalitate de combinare a datelor provenind din mai multe surse, care servește la realizarea rapoartelor, a formularelor etc. Aceste date sunt doar afișate, nu pot fi modificate (sunt read-only). Există mai multe tipuri de interogări:

- simple sau unidimensionale (*query*);
- încrucișate sau bidimensionale (*cross tab*);
- tridimensionale sau tabele pivot (*pivot table*).

a. Proiectarea interogărilor cu Query Designer

Query Designer este generatorul de interogări și reprezintă o interfață pentru realizarea interactivă a cererilor **SELECT** din SQL.

Crearea unui fișier de cereri se face interactiv, prin deschiderea ecranului de proiectare **Query Design** din meniul **File, New, Query** sau prin comanda **CREATE /MODIFY QUERY <fis.qpr>**

Va crea un fișier cu extensia **.qpr** numit *fișier de cereri*, care va putea fi executat tot prin comanda **DO <fis.qpr>**.

Mediul de lucru

Pentru utilizatorii FoxPro, interfața utilitarului Query Designer este foarte prietenoasă, punând la dispoziție o multitudine de elemente pentru realizarea operațiilor dorite: bara cu butoane, un meniu contextual, un meniu pe linia principală Query.

Bara Query Designer

- 1 Butonul Run
- 2 Butonul Add Table permite deschiderea ferestrei pentru includerea tabelor în interogare.
- 3 Butonul Remove table permite ștergerea tabelor din fereastra de proiectare a interogării.
- 4 Butonul Add Join permite adăugarea de relații temporare între tabelele interogării.
- 5 Butonul View SQL permite vizualizarea comenzii **SELECT**.
- 6 Butonul Maximize/Minimize permite modificarea dimensiunii zonei de vizualizare a tabelor.
- 7 Butonul Query Destination deschide fereastra pentru precizarea destinației interogării.



Figura 14-2: Bara Query Designer

Meniul contextual

Se deschide, așa cum știm, prin clic dreapta și conține câteva acțiuni legate de interogare, cum ar fi execuție (**Run**), adăugarea unei tabelă (**Add table**), ștergerea unei tabelă (**Remove table**), vizualizarea comenzii **SELECT** (**View SQL**), deschiderea ferestrei cu butoane pentru alegerea destinației interogării (**Output setting**).

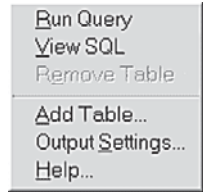


Figura 14-3:
Meniul contextual

Meniul Query

Query	Window	Help
Add Table...		
Remove Table		
Remove Join Condition		
Output Fields...		
Join...		
Filter...		
Order By...		
Group By...		
Miscellaneous...		
Query Destination...		
View SQL		
Comments...		
Run Query		Ctrl+Q

Adăugarea unei tabelă
Ștergerea unei tabelă
Ștergere legătură
Deschide tab-ul Fields – selectare câmpuri
Deschide tab-ul Join – fixare legături
Deschide tab-ul Filter – fixare filtru
Deschide tab-ul Order – fixare cheie de ordonare
Deschide tab-ul Group – fixare cheie de grupare
Deschide tab-ul Miscellaneous – alte opțiuni
Destinația se alege dintr-un ecran cu butoane
Vizualizarea comenzii SELECT generate
Introducerea comentariilor
Execuție fișier.qpr

Figura 14-4: Meniul Query

Fereastra de proiectare a unei interogări

Fereastra principală a utilitarului Query Designer este structurată în două părți: cea de sus permite vizualizarea și editarea surselor de date.

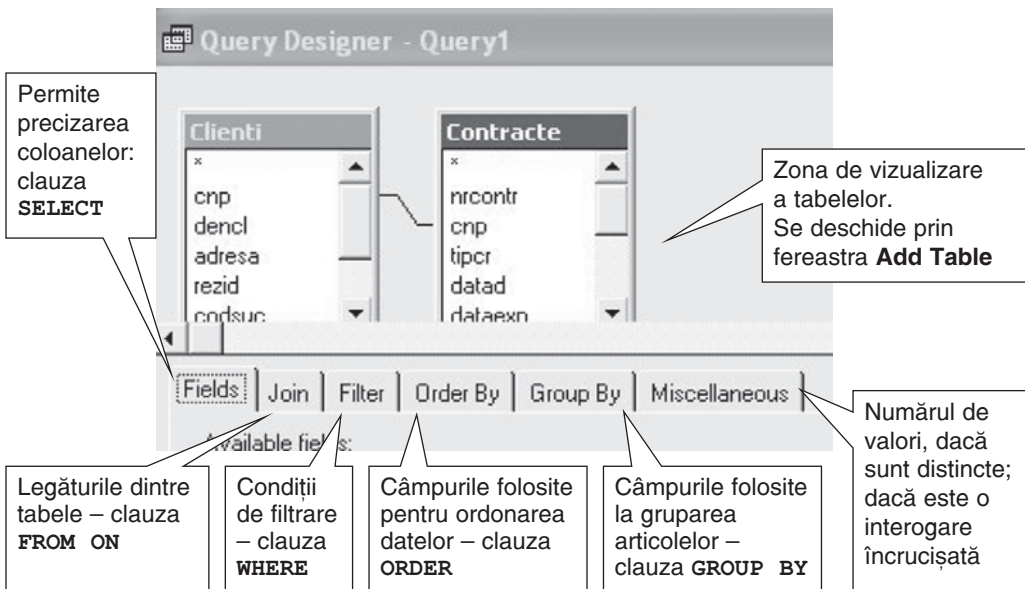


Figura 14-5: Fereastra de proiectare a interogării

Ecranul **Query Destination** permite alegerea dintre următoarele destinații:

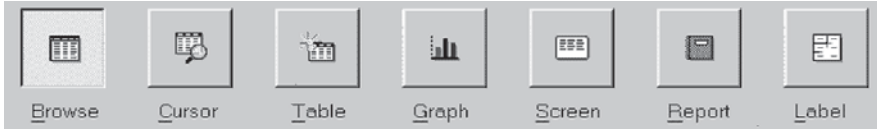


Figura 14-6: Ecranul Query Destination

Browse (afișare în fereastra Browse); **Cursor** (afișare într-o tabelă temporară pe durata sesiunii de lucru curente); **Table/DBF** (înscrierea informațiilor într-o tabelă); **Graph** (reprezentări grafice); **Screen** (afișarea datelor pe ecran); **Report** (afișare sub formă de raport); **Label** (afișare sub formă de etichetă).

Exemple

Vom explica particularitățile proiectării interogărilor prin rezolvarea pas cu pas a unor teme.

Tema 1. Pașii construirii unei interogări, tab-urile Fields, Filter, Order

Firma „**Condurul de aur**“, ce produce articole din pielărie, a cărei activitate de aprovizionare a fost analizată în lecțiile anterioare, încheie cu diverși parteneri contracte pentru vânzarea producției sale, înscrise în tabela CONTRACTE (numărul contractului, data, nume_beneficiar, nume_produș, cantitatea contractată, prețul promis la contractare). *Un contract se referă la un singur produs.* La contractare s-a estimat un anumit preț de vânzare, numit preț-promis. Produsele se facturează la livrare, evidența acestora fiind ținută în tabela FACTURI (nr_faktură, data, nr_contract, cantitatea livrată, prețul efectiv de vânzare).

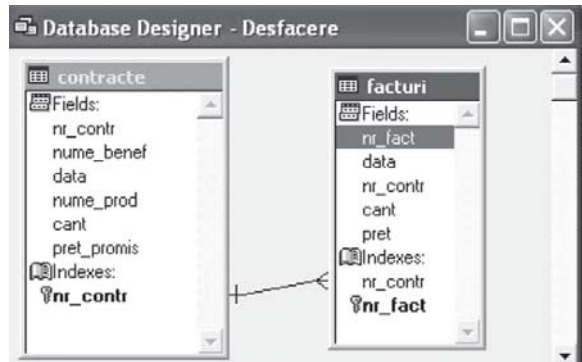


Figura 14-7: Tabelele CONTRACTE și FACTURI

Contracte					
	Nr_contr	Nume_benef	Nume_prod	Cant	Pret_promis
	121	agricola	pantofi	50	1000000
	123	intim srl	papuci	100	2000000
	122	agricola	pantofi	100	125000
	124	intim srl	pantofi	200	250000
	125	intim srl	sandale	150	175000

Facturi				
	Nr_fact	Nr_contr	Cant	Pret
	1000	122	10	100000
	1008	122	35	100000
	7412	124	10	300000
	7415	124	25	300000
	7421	124	10	300000
	1011	122	20	100000
	1025	122	30	100000

Figura 14-8: Datele din tabelele CONTRACTE și FACTURI

Presupunem că sunt create tabellele și incluse în baza de date DESFACERE.

Problemă

Ce contracte au fost încheiate cu firma „intim srl“ pentru produsul „pantofi“? Sau, altfel formulat, dorim o situație cronologică a tuturor contractelor încheiate pentru produsul „Pantofi“ de către firma „intim srl“.

Rezolvare


Pasul 1. Deschidem fereastra Query Designer selectând **File, New, Query**. Se deschide fereastra Add Table or View pentru precizarea tabelului sursă. Fereastra Add Table or View se deschide din Query Designer pentru includerea altei tabelle selectând **Query, Add Table** sau din meniul contextual. Includem tabellele Contracte.

Pasul 2. Precizăm coloanele întrebării în **tab-ul Fields**. Câmpurile tabellelor deschise în zona Table apar în lista Available Fields și pot fi mutate prin butonul ADD în lista Selected Fields, formând coloanele întrebării.

Vom selecta din câmpurile dorite din tabellele CONTRACTE: numărul contractului, produsul, cantitatea. Pentru valoarea contractului folosim zona de editare a expresiilor întrebării.

După construirea expresiei $\text{Contracte.cant} * \text{Contracte.pret_promis}$ în zona de editare vom apăsa butonul **Add** pentru includerea expresiei în lista coloanelor.

Pentru construirea comodă a expresiilor folosim Expression Builder

prin clic pe butonul .

Selectăm tabellele (lista From Table), câmpurile (lista Fields), variabilele (lista Variables) sau funcția adecvată din listele Functions. Prin dublu clic se trimite selecția în zona de editare. Verificăm corectitudinea expresiei prin clic pe butonul Verify.

Pasul 3. Fixăm un filtru prin **tag-ul Filter**. Expresia filtru se compune prin asocierea expresiilor relaționale cu operatorii logici and și or. Folosim lista derulantă a operatorilor relaționali

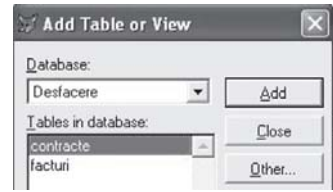


Figura 14-9: Fereastra Add Table or View

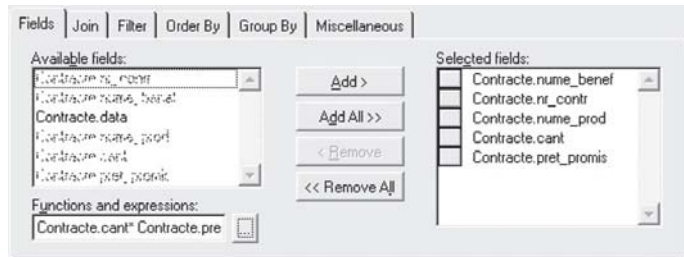


Figura 14-10: Construirea expresiei

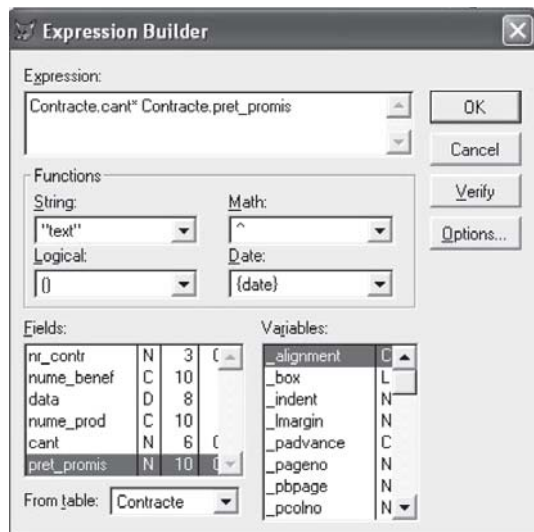


Figura 14-11: Expression Builder

(Criteria) și lista celor logici (Logical) care vor forma condiția. Nu încadrăm șirurile între ghilimele, nici datele calendaristice între apostrofuri, pentru că Query Designer asociază constantei tipul câmpului.

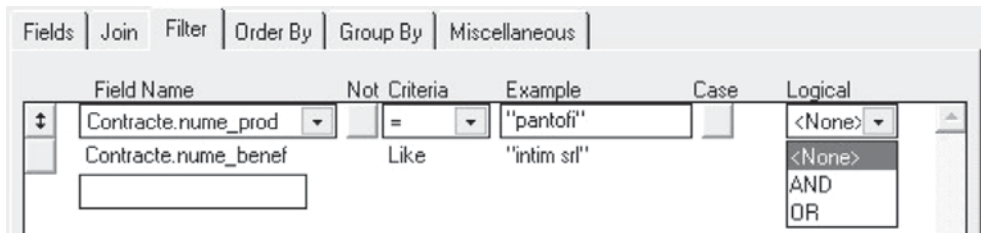


Figura 14-12: Realizarea unui filtru

Fiecare nouă expresie relațională se adaugă prin clic pe butonul Insert.

Pasul 4. Fixăm câmpurile de ordonare prin **tab-ul Order By**.

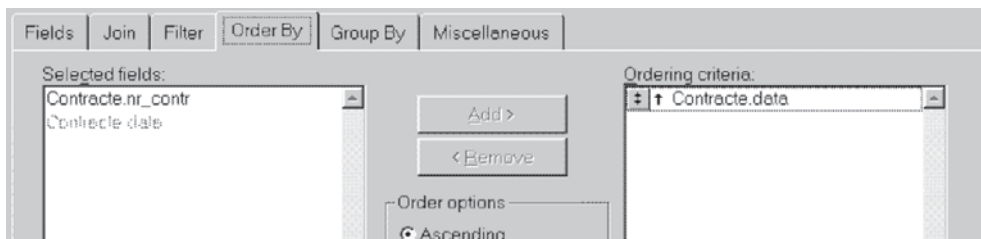


Figura 14-13: Tab-ul Order By

Pot fi alese maxim 3 câmpuri pentru ordonarea datelor. În problema noastră, vom folosi Contracte.data. Dacă trebuie introduse mai multe câmpuri, ordinea este esențială, ca și la comanda Sort. Putem inversa câmpurile în fereastra Ordering Criteria, prin tragere și plasare a butonului asociat liniei.

Pasul 5. Vizualizăm interogarea prin clic pe butonul **Run Query** 

Query					
	Nume_benef	Nr_contr	Nume_prod	Cant	Pret_promis
▶	intim srl	124	pantofi	200	250000

Figura 14-14: Vizualizarea interogării

Pasul 6. Vizualizăm comanda SQL asociată prin View SQL

```
SELECT Contracte.nume_benef, Contracte.nr_contr,
Contracte.nume_prod, ;
Contracte.cant, Contracte.pret_promis FROM
desfacere!contracte;
WHERE Contracte.nume_prod = "pantofi" AND
Contracte.nume_benef LIKE "intim srl"
```

Tema 2 Folosirea tabelor legate, grupuri, totalizări. Tab-urile Join, Group și opțiunea Query Destination

Problema

Fie baza de date DESFACERE. Obțineți într-o tabelă cu numele Centralizare o situație centralizatoare a livărilor de pantofi pe beneficiari și contracte. Pentru fiecare beneficiar și contract interesează numărul contractului, data, cantitatea contractată, numărul de facturi trimise și suma livrărilor (cantității facturate).

Observați situația dorită într-o fereastră Browse.

Query								
	Nr_contr	Nume_benef	Data	Nume_prod	Contractat	Facturat	Diferenta	Nr_facturi
	122	agricola	07/01/02	pantofi	100	95	5	4
	124	intim srl	07/01/02	pantofi	200	45	155	3

Figura 14-15: Situația centralizatoare

Rezolvare

Pasul 1. Lansăm generatorul de interogări selectând **File, New, Query, New** sau prin comanda **CREATE QUERY**.

Pasul 2. În fereastra de proiectare a interogării deschidem tablele Contracte și Facturi fie din meniul contextual, fie selectând **Query, Add Table**.

Pasul 3. Fixăm condiția de legătură în **tab-ul Join**. Observăm conservarea legăturii dintre tablele Contracte și Facturi, definită la nivelul bazei de date.

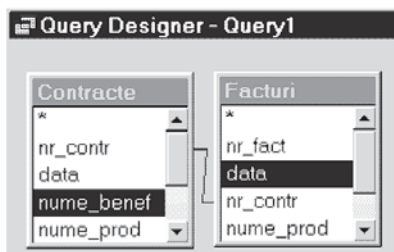


Figura 14-16: Cele două table în fereastra de proiectare

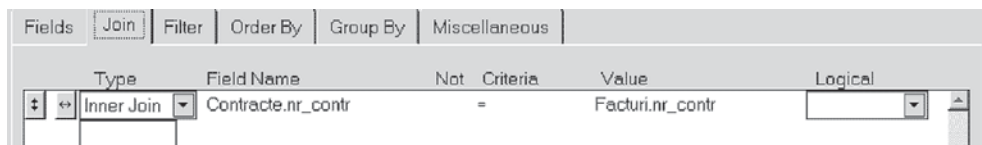


Figura 14-17: Vizualizarea legăturii în tab-ul Join

Tipuri de asocieri între tablele legate:

- *Inner Join* – numai articolele comune care satisfac criteriul;
- *Left Outer Join* – toate înregistrările din tabela din stânga (părinte) și numai cele care satisfac criteriul din tabela din dreapta (deci în stânga vom avea și articole care nu satisfac condiția de legătură!);
- *Right Outer Join* – toate articolele din dreapta relației (copil) și numai cele care satisfac criteriul din tabela din stânga;
- *Full Join* – toate articolele din ambele table (concatenare completă).

Pasul 4. Fixăm condiția de filtrare prin **tab-ul Filter**.

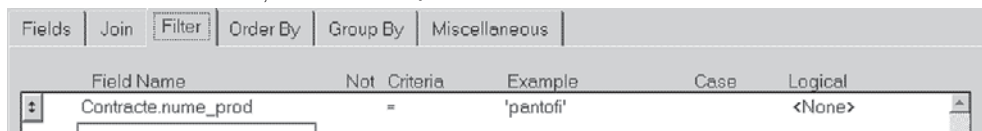


Figura 14-18: Condiția de filtrare în tab-ul Filter

Pasul 5. Fixăm câmpul folosit ca grup în **tab-ul Group By**. De obicei, se folosește gruparea, apoi se fac calcule asupra grupului. Calculele se fac prin funcțiile:

AVG, COUNT, MAX, MIN, SUM. În cazul nostru, dorim gruparea facturilor pe contracte, deci vom indica drept cheie de grupare câmpul `facturi.nr_contr`.

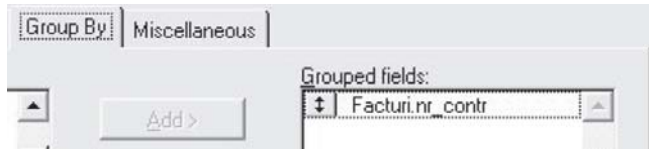


Figura 14-19: Câmpul de grupare în tab-ul Group By

În general, pentru grupuri se pot folosi funcțiile următoare:

- **COUNT ()** – numără înregistrările grupului;
- **SUM (...)** – însumează valorile pentru toate înregistrările grupului;
- **AVG (...)** – calculează media valorilor articolelor din grup;
- **MIN (...)** – extrage valoarea minimă a expresiei din grup;
- **MAX (...)** – extrage valoarea maximă din grup;
- **COUNT (DISTINCT)** – numără aparițiile grupului;
- **SUM (DISTINCT)** – însumează valorile pentru articolele distincte etc.

Pasul 6. Fixăm coloanele rezultat ale cererii în **tab-ul Fields**.

Vom selecta și vom muta prin butonul **ADD** câmpurile `nr_contract`, `nume_benef`, `data`. Pentru cantitatea contractată, dorim asocierea unui alt nume, deci vom lansa **Expression builder**, iar în zona de editare vom plasa pentru câmp aliasul `Contractat`.

În problema noastră se cere o grupare a facturilor pe fiecare contract. Pentru suma cantităților facturate folosim expresia `Sum(Facturi.cant) as facturat`.

Pentru numărul de facturi vom construi o expresie folosind funcția `COUNT(facturi.nr_fact)`, iar pentru diferență vom scrie expresia `Contracte.cant-sum(facturi.cant) as diferenta`.

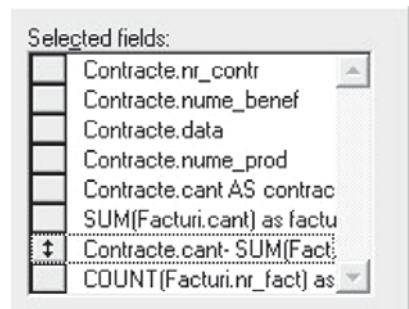


Figura 14-20: Fixarea coloanelor rezultat în tab-ul Fields

Pasul 7. Alegem **destinația finală** a rezultatelor interogării prin opțiunea **Query Destination**. Alegem opțiunea **Table**.

Pasul 8. Executăm interogarea folosind butonul sau comanda **Run query**.

Pasul 9. Vizualizăm comanda **SQL** prin **View SQL**.

```
SELECT Contracte.nr_contr, Contracte.nume_benef, Contracte.data, ;
Contracte.nume_prod, Contracte.cant AS contractat, ;
SUM(Facturi.cant) AS facturat, ;
Contracte.cant-SUM(Facturi.cant) AS diferenta, ;
COUNT(Facturi.nr_fact) AS nr_facturi;
FROM desfacere!contracte INNER JOIN desfacere!facturi ;
ON Contracte.nr_contr = Facturi.nr_contr;
WHERE Contracte.nume_prod = "pantofi";
GROUP BY Facturi.nr_contr INTO TABLE livrari.dbf
```

Tema 3 Folosirea interogărilor pentru obținerea informațiilor distincte. Prezentarea tab-ului Miscellaneous

Problema

Fie baza de date DESFACERE. Afișați partenerii de afaceri care au încheiat contracte cu firma „Condurul de aur” în luna februarie.

Rezolvare

Pasul 1. Lansăm **Query Designer** și includem tabela **Contracte**.

Pasul 2. Precizăm coloanele în tab-ul **Fields**: nume beneficiar.

Pasul 3. Fixăm condiția filtru. Pentru a filtra după o expresie este necesară alegerea opțiunii **Expression** din lista **Field Name**. Deschidem **Expression Builder**, pentru a construi expresia.

Pasul 4. Fixăm opțiunea

NO Duplicates în tab-ul **Miscellaneous**. Din acest tab este posibilă și stabilirea unui număr sau procent de vizualizat din rezultatul interogării (caseta **Number of records** sau comutatorul **Percent**).

Pasul 5. Salvăm, apoi închidem **Query Designer**.

Pasul 6. Executăm prin comanda **DO fis.qpr**.

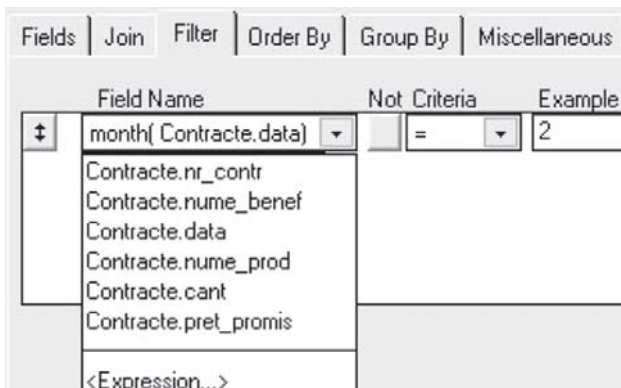


Figura 14-20: Fixarea condiției pentru filtru

Tema 4 Prezentarea rezultatelor unei interogări sub forma unei diagrame

Problema

Fie baza de date DESFACERE. Afișați variația valorică a contractelor încheiate de firma „intim srl” pe fiecare produs.

Rezolvare

Pasul 1. Proiectăm o interogare prin **Query Designer**, precizând drept sursă tabela **Contracte** și coloanele: nume beneficiar, nume produs și expresia pentru valoare ($\text{cant} \times \text{pret}$).

Pasul 2. Fixăm pentru grupare câmpurile nume beneficiar și nume produs.

Pasul 3. Fixăm drept filtru expresia $\text{Contracte.num_benef} = \text{„intim srl”}$.

Pasul 4. Precizăm destinația ca raport grafic: **Query, Query Destination, Graph..**

Pasul 5. Executăm interogarea prin clic pe butonul **Run** și observăm deschiderea ferestrelor de proiectare a graficului prin **Graph Wizard**.

În fereastra **Graph Wizard** indicăm variabilele pentru axe prin tragere și plasare de la câmpul nume_prod la axa O_x și de la Valoare la Data Series .

Observație: Atributele graficului pot fi schimbate prin dublu clic pe fereastra finală a Graph Wizard. Se deschide fereastra de editare a graficului, iar pe meniul principal pot fi folosite opțiunile **Chart, Tools, Data** etc. Dacă butonul **Graph** nu este activ, putem salva interogarea într-o tabelă, apoi, după ieșirea din generatorul de interogări, selectăm **Tools, Graph Wizard** și alegem tabela anterior generată.

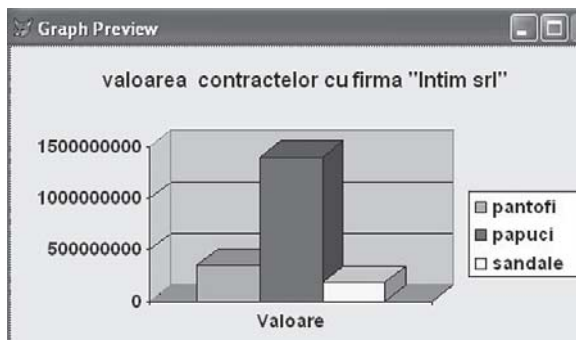


Figura 14-21: Reprezentarea grafică a rezultatelor

Pasul 6. Vizualizăm comanda Select prin View SQL:

```
SELECT Contracte.ume_benef, Contracte.ume_prod, ;
      sum(Contracte.cant* Contracte.pret_promis) as valoare;
FROM desfacere!contracte, desfacere!facturi;
WHERE Contracte.ume_benef = "intim srl";
GROUP BY Contracte.ume_benef, Contracte.ume_prod;
INTO CURSOR SYS(2015)
DO (_GENGRAPH) WITH 'QUERY'
```

b. Proiectarea rapidă a interogărilor cu Query Wizard

Pentru proiectarea rapidă a interogărilor, folosim aplicația Query Wizard, care poate fi lansată selectând **Tools, Wizard, Query** sau **File, New, Query** și alegând aplicația wizard. În fereastra de dialog deschisă se poate opta pentru tipul de interogare dorit.

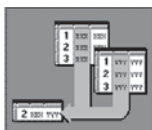
Exemple

Tema 5 Folosirea aplicației Query Wizard

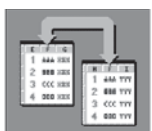
Problema

Obțineți situația contractelor, ordonată pe beneficiari, care au facturări după 1 ianuarie 2002.

Vom afișa de fapt informații despre contracte, pentru fiecare contract informații despre facturile trimise în contul acestuia(câmpul; Nr_fact și data).

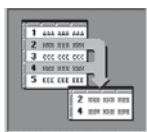


Pasul 1. Selectăm câmpurile care vor forma ieșirea din interogare. Atenție, sunt câmpuri din tabele, nu expresii. Dacă dorim folosirea expresiilor, edităm fișierul .Qpr creat cu aplicația Wizard în Query Designer.



Pasul 2. Dacă sunt mai multe tabele, specificăm relația. În cazul nostru scriem:

```
Contracte.nr_contr= facturi.nr_contr
```



Pasul 3. Filtrăm datele rezultate printr-o expresie logică formată din mai multe expresii relaționale. În cazul nostru scriem:

Facturi.data greater then or equal 01/01/02



Pasul 4. Fixăm câmpurile de ordonare, în cazul nostru, **contracte.nume_benef**. La fel, numărul de câmpuri este limitat la 3. Ordinea plasării acestor câmpuri în lista Available Fields contează pentru că dă prioritatea cheii la sortare.



Pasul 5. Salvăm ca fișier .Qpr; putem rula imediat fișierul (opțiunea **Save and run**) sau putem intra în Query Designer (opțiunea **Save query and modify it in Query Designer**).

Proiectarea fișierelor View

Spre deosebire de interogări (care sunt de tip Read only) **vederile, perspectivele sau imaginile** (views) sunt fișiere care se pot modifica și, mai mult, transmit aceste corecții tabelului sursă din care sunt create.

Vederile sunt de două tipuri: locale (create din tabele sau alte vederi locale) sau la distanță (create din tabele situate pe serverul ODBC aflat la distanță).

Fișierele sunt memorate într-o bază de date și pot fi gestionate ca și celelate tabele de date. Acest lucru este un avantaj, dar atenție: nu putem utiliza vederea decât dacă baza de date este deschisă.

Când folosim vederea, definiția ei va crea o comandă SQL care va da setul de date pe care îl putem folosi. Realizarea fișierelor vedere (având extensia .Vue) se poate face interactiv cu utilitarul View Designer sau cu asistenții Wizard.

a. Proiectarea fișierelor vedere prin View Designer

Apelul utilitarului pentru crearea unei vederi se face fie direct, selectând **File, New, View**, fie prin tastarea comenzii **CREATE VIEW** în fereastra de comenzi. Meniul asociat, ca și fereastra de proiectare, sunt asemănătoare celor de la Query Designer.

Exemple

Tema 6 Etapele și specificul operației de creare a unui fișier View

Problema

Fie baza de date DESFACERE. Creați o vedere cu beneficiarii unui produs X. Va fi permisă editarea câmpurilor, cu excepția cheii de legătură (**nr_contr**) și a numelui beneficiarului (**nume_benef**).

Rezolvare

1. Deschidem baza de date Contracte prin **OPEN DATABASE**.

ATENȚIE! Vederile pot fi create numai în contextul unei baze de date deschise, unde de fapt, sunt și memorate.

2. Lansăm utilitarul View Designer, selectând **File, New, View, New**.
3. Ecranul de proiectare este cunoscut de la Query Designer. Deschidem tabelele Contracte și Facturi prin opțiunea **Add Table** din meniul principal, numit tot **query**.
4. Prin Tab-ul **Fields** selectăm câmpurile care vor forma coloanele tabelului view: **Contracte.nr_contr**, **contracte.data**, **contracte.nume_benef**, **facturi.nr_fact**, **facturi.data**.

ATENȚIE! Deși câmpurile Memo pot fi incluse într-o vedere, ele nu pot fi actualizate.

5. Prin tab-ul **Join** fixăm legătura contracte→1-n→facturi.
6. Stabilim cheile de ordonare, grupate la fel ca la Query Designer.
7. Introducem condiția de filtrare. Până acum am folosit tab-ul **Filter** punând în membrul drept al condiției o constantă. În problemă dorim să filtrăm vederea prin introducerea unei variabile, astfel încât la execuție să obținem rezultate diferite.

Pentru introducerea variabilei în filtru, îi precedăm numele cu semnul întrebării. Exemplu: **facturi.nume_prod=?x**. Apoi definim numele variabilei și tipul ei în fereastra View Parameters, deschisă prin **Query, View Parameters**. La execuție va fi cerută valoarea acestei variabile.

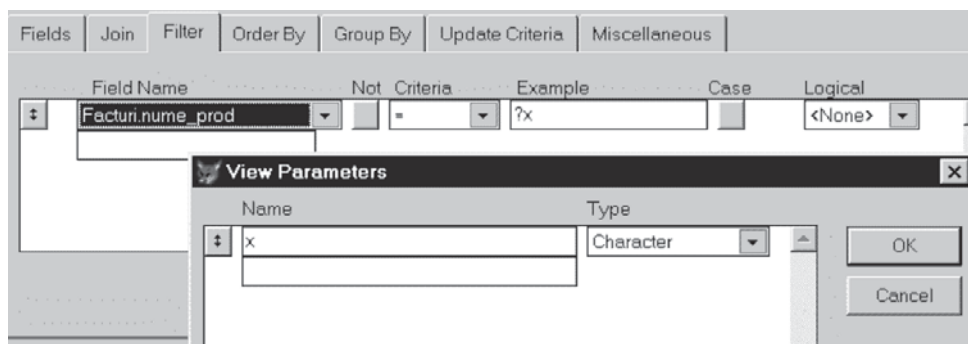


Figura 14-22: Introducerea variabilei în filtru

Tab-ul **Update Criteria** este folosit pentru actualizarea datelor. În fereastra sa principală sunt afișate toate coloanele precizate ca ieșire din vedere. Lateral sunt doi indicatori: primul (cheia) arată dacă este câmp cheie într-o tabelă sursă și al doilea (creionul) dacă poate fi actualizată coloana prin vedere.

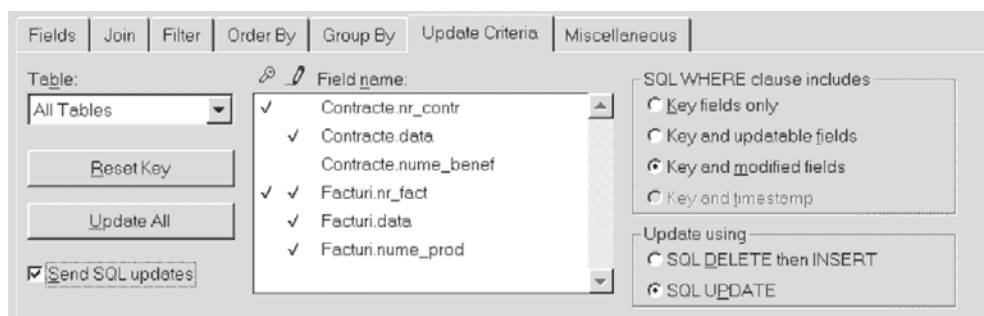


Figura 14-23: Tab-ul Update Criteria

Trebuie știut că:

- Un câmp cheie nu poate fi modificat. Dacă am stabilit deja niște câmpuri cheie diferite de cele din tabela inițială, alegem **Reset Key** și sistemul va căuta în tabela sursă câmpurile cheie după indecșii primari existenți și va permite fixarea altuia drept cheie.
- Vom modifica doar câmpurile marcate pentru această operație. În fereastra Update observăm că numele beneficiarilor nu poate fi modificat.
- Schimbarea setării inițiale a indicatorilor asociați coloanelor vederii se face prin poziționarea pe numele câmpului și selectarea comutatorului corespunzător (primul pentru fixarea cheii, al doilea pentru actualizare).
- Comutatorul **Send SQL Updates** trebuie activat pentru a folosi vederea la actualizarea datelor din surse.
- Partea dreaptă a ferestrei Update (SQL WHERE) este folosită pentru vederi la distanță și o vom discuta la lecția următoare.



Figura 14-24: Câmpurile marcate pot fi modificate

8. Salvăm și ieșim din proiectare.

Exemple

Tema 7 Proiectarea interogărilor încrucișate cu Crosstab Wizard

O **interogare încrucișată** este rezultatul unei interogări speciale, care permite analizarea relației dintre un câmp al tabelului de date și alt câmp al aceluiași tabel.

Problema

Fie baza de date DESFACERE. Afișați cantitatea totală vândută din fiecare produs în fiecare lună din anul curent. Situația va fi afișată sub forma unei matrici: pe linii avem produsele, iar pe coloane avem lunile. La fiecare produs, calculați totalul vânzărilor.

Observați rezultatul obținut printr-o interogare încrucișată pornind de la tabela Facturi, dată în continuare.

Rezolvare

Pasul 1. Pregătirea sursei de date pentru Crosstab Wizard.

Observați tabela Facturi. Prima problemă pe care o avem de rezolvat este asocierea numelui de produs pentru fiecare factură. Știind că o factură se referă la un anumit contract și un contract se semnează pentru un produs, este suficientă legătura între tabelele Contracte și Facturi. De exemplu, contractele 122 sunt pentru „pantofi“, 123 pentru „papuci“, iar 125 pentru „sandale“. O altă problemă care trebuie rezolvată înaintea proiectării interogării Crosstab este identificarea lunii din data facturii.

	Nr_fact	Data	Nr_contr	Cant	Pret
	7412	03/08/02	124	10	300000
	7415	03/11/02	124	25	300000
	7421	05/15/02	124	10	300000
	1011	05/25/02	122	20	100000
	1025	05/26/02	122	30	100000
	1026	05/12/02	125	25	150000
	1027	06/02/02	125	50	150000
	1028	04/02/02	123	25	150000
	1029	04/15/02	123	10	150000

Figura 14-25: Tabela Facturi

	Nume_prod	N_2	N_3	N_4	N_5	N_6	Total
	pantofi	200	400	.NULL.	400	.NULL.	1000
	papuci	.NULL.	.NULL.	200	.NULL.	.NULL.	200
	sandale	.NULL.	.NULL.	.NULL.	150	150	300

Figura 14-26: Contractele pentru pantofi, papuci, sandale

Deci, creăm un fișier vedere prin **File, New, View**.

Adăgăm tabelele Contracte și Facturi și indicăm drept coloane denumirea produsului din tabela Contracte, iar cantitatea din tabela Facturi. De asemenea, folosim funcția **Month(facturi.data)** pentru a totaliza vânzările pe luni. Salvăm într-un fișier View cu numele View1.

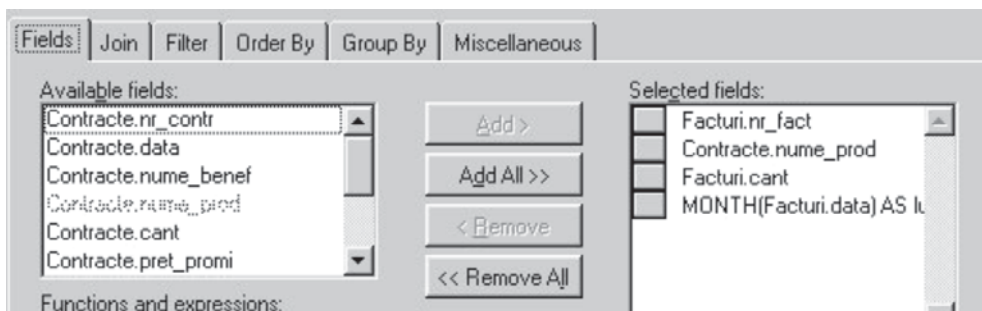


Figura 14-27: Crearea fișierului vedere

Pasul 2. Lansăm Cross Tab Wizard selectând **Tools, Wizard, Query, Cross-Tab Wizard**.

- Selectăm cele trei necesare: luna, numele produsului, cantitatea din fișierul cursor.
- Completăm axele: dorim pe linii produsele (sunt multe), pe coloane trecem lunile, iar în celulele matricii trecem cantitatea.
- Specificăm operația de calcul pe care o va face utilitarul (**SUM**).

În mod implicit, în ultima coloană se face suma, dar putem solicita și alte calcule, cum ar fi: numărarea celulelor care conțin date (de exemplu, numărul de luni în care a fost solicitat fiecare produs) sau calcularea procentajului (cât la sută din totalul vânzărilor reprezintă fiecare produs) etc.

- Salvăm ca fișier .qpr; acesta poate fi rulat imediat (opțiunea **Save and run**) sau putem intra în generatorul de interogări pentru modificări.

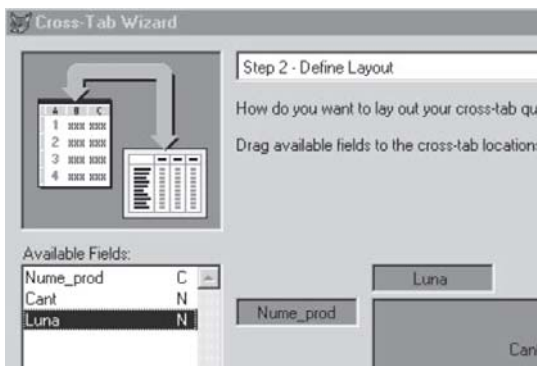


Figura 14-28: Stabilirea layout-ului

Tema 8 O problemă recapitulativă

Stabilirea necesarului de aprovizionat folosind Query Design

Realizați prin utilitarul Query Design operația de calcul a necesarului de aprovizionat (din aplicația anterioară). Baza de date este Aproviz și cuprinde tabelele Plan → 1,n consum → 1,n stoc.

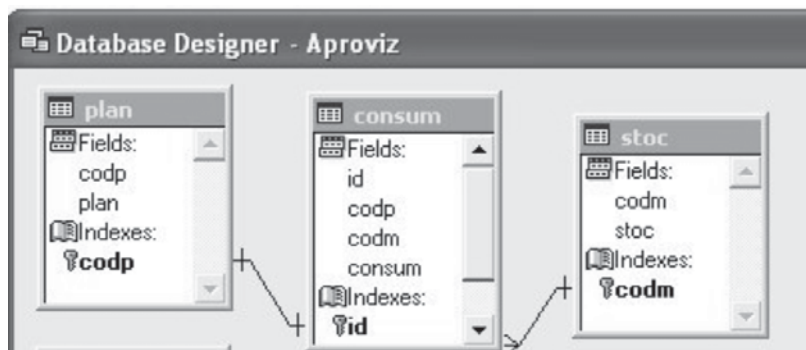


Figura 14-29: Baza de date Aproviz

Datele de plecare sunt memorate în tabelele Plan, Consum și Stoc.

Plan	Codp	Plan
	1	100
	2	200

Consum	Id	Codp	Codm	Consum
	1	1	1	5
	2	1	2	2
	3	1	3	3
	4	2	2	4

Stoc	Codm	Stoc
	1	1000
	2	5000
	3	0
	4	4000

Figura 14-30: Datele de pornire din cele trei tabele

Pasul 1. Calculăm produsul cantității planificate cu unitatea de consum ca să aflăm necesarul de material pentru realizarea planului.

Lansăm View Designer și indicăm tabelele Plan și Consum. Precizăm coloanele, iar printre ele **plan.plan*consum.consum as necplan** pentru fiecare linie din consum legată de plan prin **codp**. Vom denumi fișierul vedere **necplan**.

Observați fereastra de proiectare pentru **necplan.vue**

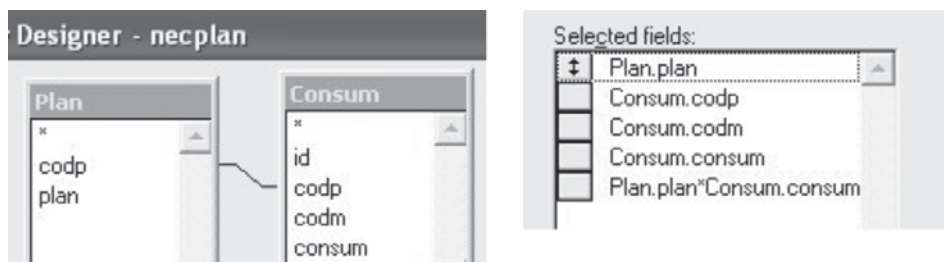
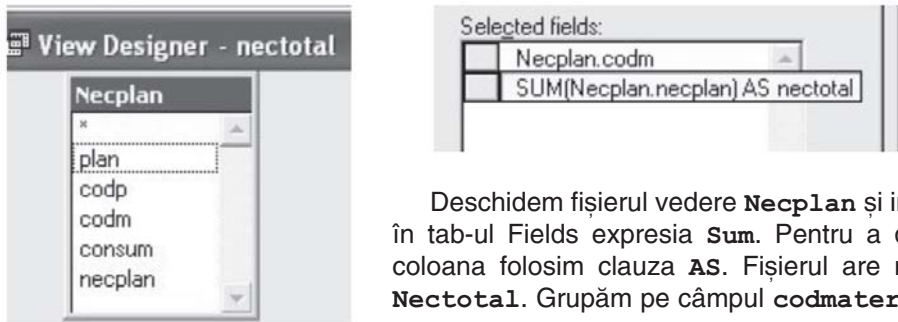


Figura 14-31: Fereastra de proiectare pentru **necplan.vue**

Pasul 2. Folosim tabela virtuală creată anterior. Pentru fiecare `codmaterial` din vederea `necplan` creată calculăm suma.



Deschidem fișierul vedere `Necplan` și indicăm în tab-ul Fields expresia `Sum`. Pentru a denumi coloana folosim clauza `AS`. Fișierul are numele `Nectotal`. Grupăm pe câmpul `codmaterial`.

Figura 14-32: Fereastra de proiectare pentru `nectotal.vue`

Pasul 3. Aflăm necesarul de aprovizionat la fiecare material prin diferența stoc-necesar total. Noul fișier vedere folosește vederea `Nectotal` creată anterior și tabela `Stoc`. Relația trebuie fixată pe câmpul `codm`.

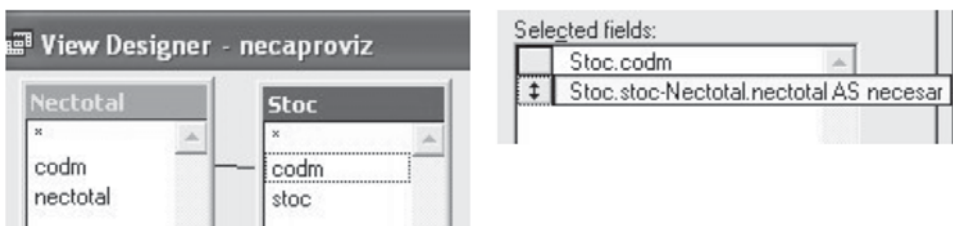


Figura 14-33: Aflarea necesarului de aprovizionat

Necplan					Necaproviz		Nectotal	
Plan	Codp	Codm	Consum	Necplan	Codm	Necesar	Codm	Nectotal
100	1	1	5	500	1	500	1	500
100	1	2	2	200	2	4000	2	1000
200	2	2	4	800	3	-300	3	300
100	1	3	3	300				

Figura 14-34: Tabelele intermediare rezultate în proiectare



sarcini de laborator

I. Considerăm o bază de date cu informații legate de salarizarea personalului într-o firmă. Tabelele sunt:

- PERSONAL (marca sau codul persoanei, numele, funcția, locul de muncă)
- Salar (marca, salariul de bază);
- CAR (marca, cotizația-lunară, suma-împrumutată, data împrumutului, suma lunară pentru lichidarea împrumutului, suma totală achitată din împrumut, fondul acumulat din cotizațiile lunare anterioare),
- RATE (marca, valoarea totală a cumpărăturilor prin rate, banca și contul la bancă unde se plătesc ratele, suma plătită lunar, numărul de rate, valoarea ultimei rate).

Se cere:

1. Proiectați baza de date și populați tabelele cu date de test.
2. Aflați prin comenzi SQL:
 - a) toți salariații ordonați alfabetic;
 - b) numele și locul de muncă pentru persoanele care nu plătesc impozit;
 - c) salariul mediu pe locuri de muncă;
 - d) salariul mediu pe funcții;
 - e) fondul total de salarii;
 - f) numărul de salariați pe locuri de muncă;
 - g) impozitul total, reținerile totale, totalul sporurilor pe locuri de muncă;
 - h) funcția și numele persoanei cu salariul maxim;
 - i) locurile de muncă (distincte!).
3. Folosind Query Designer, Query Wizard, View Designer, View Wizard aflați:
 - a) Câte persoane nu au împrumuturi la CAR (plătesc doar cotizația)?
 - b) Care este suma totală achitată pentru ratele contractate de o persoană X?
 - c) Care sunt persoanele care au contractat cel mai mare împrumut la CAR și au suma ratelor peste 1 milion?
 - d) Care sunt persoanele care au terminat de achitat ratele luna aceasta?
 - e) Afișați sub formă de fluturași informațiile despre salariul de bază, nume, funcție, sumele reținute la CAR și rate.
 - f) Numărați pe funcții câte persoane au rate.
 - g) Calculați pe locuri de muncă suma CAR care trebuie depusă luna curentă.
 - h) Calculați pentru fiecare bancă sumele care trebuie virate ca urmare a achitării ratelor de către salariați.
 - i) Afișați într-un raport informațiile nume, funcție, salariu de bază, total rețineri (rate+CAR), rest de plată, pe locuri de muncă, alfabetic.
 - j) Afișați sub forma unui grafic variația salariului de bază pe funcții.
 - k) Afișați sub forma unui grafic variația reținerilor totale pe locurile de muncă.
 - l) Afișați sub forma unui tabel sumele datorate de fiecare persoană ca rate_CAR+cotizație_lunară+rate_cumpărături pe fiecare lună a primului trimestru și total.

II. Știind că se ține evidența participanților la faza națională a olimpiadelor școlare încă din anul 1990, aflați:

1. La ce olimpiade au obținut premii sau mențiuni reprezentanții județului X?
2. Realizați o situație centralizatoare cu numărul total de premii pe județe și discipline în anul curent.
3. Realizați o clasificare a județelor în funcție de rezultate, acordând un punctaj fiecărui premiu (vor fi luate în ordine premiul 1, premiul 2, premiul 3, mențiune 1, mențiune 2 și mențiune 3).
4. Care sunt participanții din Cluj la olimpiada de matematică în anul curent?
5. Care este județul cu cele mai multe premii 1 în anul x?
6. La ce olimpiade s-au obținut cele mai multe premii?
7. Lista participanților la olimpiada internațională de informatică din anul trecut.
8. Există vreun județ care nu a obținut nici un premiu anul acesta?
9. Lipsește vreun an din evidențe?
10. Care sunt disciplinele la care se organizează olimpiade? Salvați în altă tabelă aceste discipline. Fixați o legătură între tabelele DISCIPLINE și OLIMPIADE și aflați care sunt anii în care nu s-a organizat concurs pentru fiecare disciplină.

III. Deschideți baza de date pentru activitatea de BIBLIOTECA. Proiectați interogările următoare prin comenzile **SELECT** și interactiv, cu utilitarele Query Design și Query Wizard:

1. Care sunt restanțierii ?
2. Cine a împrumutat anul acesta cartea X?
3. Care sunt elevii din clasa 11b înscriși la bibliotecă?



sarcini pentru realizarea unui mini-proiect

Imaginați interogări legate de activitatea specifică proiectului fiecărei echipe pentru:

- a. folosirea unei singure tabele, cu filtrarea datelor;
- b. folosirea unei singure tabele cu operații totalizatoare la nivelul unui grup;
- c. folosirea a două tabele legate;
- d. folosirea unei interogări cu parametru (View);
- e. folosirea unei vederi pentru actualizarea datelor din două tabele legate.

Comunicarea aplicației Visual FoxPro cu alte aplicații

- *Accesarea datelor la distanță; View Designer, Remote View Wizard*
- *Publicarea datelor pe Internet; Web Publishing Wizard*
- *Proiectarea paginii Web pentru căutare; WWW Search Page Wizard*
- *Proiectarea documentelor pentru e-mail*

O aplicație nu folosește totdeauna doar date situate pe calculatorul gazdă. Ea poate și trebuie să acceseze și să vadă datele situate pe alte calculatoare, chiar în alte formate decât FoxPro.

De asemenea, este util uneori să facem cunoscute anumite date publicului larg, pentru reclamă sau alte activități. Și cum internetul este cea mai eficientă cale de comunicare, ne vom ocupa de proiectarea documentelor web pentru vizualizarea informațiilor din baza de date FoxPro pe internet.

O altă activitate legată tot de comunicarea cu exteriorul este posibilitatea oferirii unor informații la cererea clienților de pe internet.

Proiectarea fișierelor vedere cu date la distanță

Pentru a accesa date situate la distanță trebuie utilizată interfața ODBC (Open DataBase Connectivity). Această interfață utilizează drivere pentru conversia sintaxei SQL de la un produs la altul.

Pași:

1. Stabilirea unei conexiuni la distanță cu altă bază de date: deschidem Control Panel și selectăm pictograma ODBC pentru a vedea ce surse de date sunt disponibile. FoxPro posedă drivere ODBC pentru partajarea datelor, printre care Access, Sybase SQL, DB2, Oracle, Paradox. Toate driverele au ecrane de configurare în care se pot specifica informații, cum ar fi versiunile, tabelele și directoarele supuse partajării.
2. Deschiderea FoxPro.
3. Deschiderea unei baze de date: Visual FoxPro stochează informațiile referitoare la conexiuni și vederi la distanță în fișierul .DBC.
4. Crearea conexiunii: selectăm **File, New, Connectivity**. Apare o fereastră pentru identificarea sursei de date, definirea numelui utilizatorului, introducerea parolei etc. Dăm un nume conexiunii. Putem defini atâtea conexiuni câte surse de date (tabele) avem în sistem. Fiecare conexiune poate susține vederi ale datelor de la distanță.
5. Lansarea generatorului de vederi prin **File, New, Remote View, New**. Apare o casetă de dialog pentru selectarea conexiunii și a sursei de date (baza de date).

6. Adăugarea tabelelor necesare din baza de date: fiecare nouă tabelă trebuie să aibă o legătură cu cele existente deja, altfel FoxPro face produsul cartezian al articolelor celor două tabele.
7. Adăugarea criteriilor de selecție prin tab-ul Join: nu se pot lega între ele tabele aparținând unor conexiuni diferite. În schimb, odată creată vederea, se poate crea o interogare sau un formular care să utilizeze date de la mai multe surse.

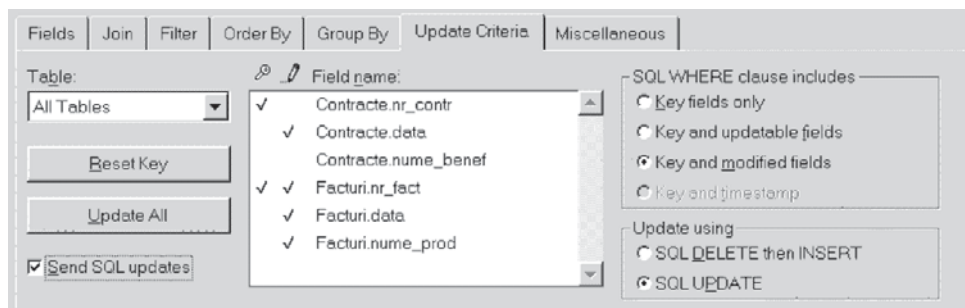


Figura 15-1: Adăugarea criteriilor

8. Filtrarea, stabilirea ordinii, a grupării, a criteriilor de actualizare, ca și în cazul unei vederi locale.
9. Precizarea condițiilor de acceptare a actualizărilor prin Update Criteria: pot exista conflicte de actualizare, ținând cont că în condițiile existenței mai multor utilizatori, aceeași tabelă poate fi solicitată la actualizare simultan de la mai multe calculatoare. Clauzele SQL **WHERE** ne ajută în această privință:
 - Key fields only – se efectuează actualizările atâta vreme cât câmpul cheie al tabelii sursă nu s-a modificat;
 - Key and updatable fields – interzice actualizarea atunci când oricare dintre câmpurile marcate pentru actualizare și-a schimbat valoarea în tabela de la distanță;
 - Key and modified fields – interzice actualizarea dacă pentru un câmp a cărui valoare s-a modificat local, valoarea acestuia se schimbă și în tabela sursă;
 - Key and timestamp – interzice actualizarea dacă valoarea cheii sau a mărcii temporare a fost modificată în tabela inițială.
 - Modificarea câmpurilor se poate face fie prin corecții directe în articolele vederii (butonul radio **SQL Update**), fie prin ștergerea înregistrărilor care au suferit corecții și reinserarea lor în varianta nouă (**butonul SQL Delete then INSERT**).

Proiectarea rapidă a vederilor cu Remote View Wizard

Pentru realizarea unei vederi cu surse de date diferite de cele ale Visual FoxPro se poate folosi și instrumentul Remote View Wizard, care desfășoară aceleași ferestre de dialog ca și Local View Wizard, cu excepția primului pas, când se indică legătura cu sursa externă. Fișierul vedere nu permite în mod automat actualizarea datelor, ci va trebui ca după salvare să-l modificăm cu View Designer, tab-ul Update Criteria.

Atenție! Fiecare vedere la distanță poate accesa o singură sursă de date. Pentru a combina un fișier Paradox cu unul FoxPro trebuie create două vederi ale celor două tabele. Apoi, presupunând că există un câmp care permite definirea unei relații, se pot combina cele două vederi în mediul de date Data Environment al unui formular sau al unui raport.

Proiectarea paginilor web pentru vizualizarea datelor pe internet

Utilitarul Web Publishing Wizard permite utilizatorilor externi vizualizarea unei baze de date. Este lansat selectând **Tools, Wizards, Web Publishing**.



Pasul 1. Selectăm baza de date sau tabela ale cărei informații se pot vizualiza pe internet. Desigur, dacă este nevoie să asigurăm filtrarea articolelor, realizăm un fișier vedere care se deschide la acest pas. Selectăm câmpurile care vor fi afișate.



Pasul 2. Precizăm ordinea de afișare a datelor indicând maxim 3 câmpuri de sortare și sensul operației.

Pasul 3. Setăm caracteristicile de design ale paginii web. Alegem modalitatea de afișare a datelor selectate; pe linii, sub formă de etichetă sau pe coloane etc.

Stilul folosit pentru formatul paginii web poate fi selectat prin comutatoarele din lista Visual Styles.

Pentru fiecare configurare a paginii alese există posibilitatea de previzualizare.

Pasul 4. Salvăm documentul. Putem salva documentul ca pagină web pentru folosire ulterioară; putem salva și lansa un editor de texte în vederea modificării paginii; putem salva și deschide într-un browser sau putem să creăm un program (cu extensia .prg) generator de pagini de web.

Isdn	Autor	Titlu	Editura	Colectia	Nrpag	Pret	Editia
2	Godfrey Devereux	Elemente de Yoga	Teora	Interzone	128	75000	1999
1	Peter Drucker	Realitatile lumii de maine	Teora	Economie	200	50000	1999
3	Kevin Marlove	Utilizare Access '97	Teora	Calculator	328	60000	1998

Figura 15-2: Exemplu de format de afișare a unei colecții de cărți

Proiectarea paginilor de căutare pe internet

În vederea căutării anumitor informații de către vizitatorii internetului, Visual FoxPro dispune de utilitarul WWW Search Page Wizard, lansat selectând **Tools, Wizards, All, WWW Search**.

Pasul 1. Selectăm tabela sau baza de date în care vor fi căutate datele. Este necesar ca tabela aleasă să aibă cel puțin o cheie principală. În cazul nostru, luăm tabela Carti cu cheia principală ISBN.

Pasul 2. Precizăm indexul după care se face căutarea. În cazul nostru, este colecția.

Pasul 3. Introducem textele pentru titlul paginii (Search Page Title) și indicațiile pentru căutare (Search Page Description).

Pasul 4. Introducem o imagine de fundal (Background Image) și o imagine de antet a paginii (Header Image). Fișierele imagine au extensiile .gif și .jpg. Comutatorul **Provide Ability to Download the Result Set as a File** permite trimiterea rezultatului căutării sub forma unui fișier.

Pasul 5. Selectăm câmpurile care vor fi incluse în pagina de rezultate. Numărul maxim de câmpuri selectate este 5. Să presupunem că dăm acces la titlu, autori, editură, ediție, număr de pagini.

Pasul 6. Precizăm imaginile de fundal și antet pentru pagina rezultat al căutării. De asemenea, setăm numărul maxim de înregistrări returnate în căutare. În zona de editare ODBC Data Source introducem sursele de date ODBC create în sistem pentru a face accesibile tabelele FoxPro serverului internet (implicit Microsoft).

Pasul 7: Salvăm pagina web introducând numele și directorul unde sunt memorate cele trei fișiere pe care le creează utilitarul. Fișierele au extensiile .htm – pagina HTML folosită pentru căutare, .htx – fișierul rezultat al căutării și .idc – fișier cu instrucțiunea SQL Select prin care se realizează căutarea. Vom crea directorul carti și vom salva fișierele tot sub numele Carti.

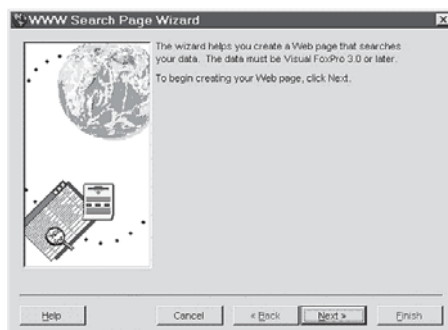


Figura 15-3: Aplicația WWW Search Page Wizard



Figura 15-4: Pagină de căutare a cărților dintr-un depozit denumit „Cartea prin poștă“

Proiectarea și transmiterea documentelor prin e-mail.

Utilitarul Mail Merge Wizard

Visual FoxPro permite transmiterea directă prin e-mail a unor documente ca etichete, scrisori, cataloage.

Utilitarul Mail Merge Wizard preia datele din tabelele Visual FoxPro specificate și lansează instrumentul Word pentru compunerea documentului de expediat. Odată realizat documentul, poate fi trimis imediat la adresele dorite. Modificările ulterioare din baza de date sunt reflectate automat în document. Utilitarul se lansează selectând **Tools, Wizard, Mail Merge**.

Exemplu

Ne propunem să trimitem tuturor prietenilor noștri invitații de nuntă. În tabela Agenda, avem numele și adresele lor de e-mail.

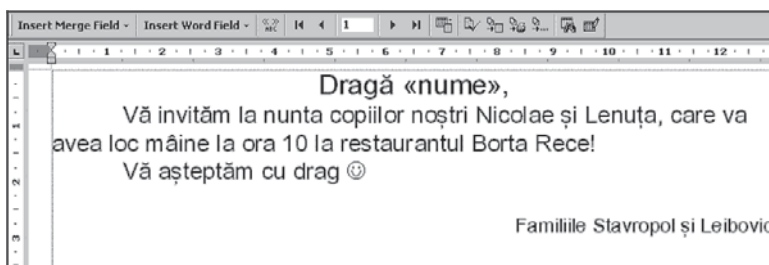


Figura 15-5: Scrierea documentului de expediat

În vederea elaborării scrisorii și trimiterii prin e-mail, vom lansa utilitarul Mail Merge Wizard.



Pasul 1. Selectăm câmpurile din tabela Agenda: numele și adresa de e-mail. Atenție, nu putem folosi câmpuri din mai multe tabele. In aceste situații va trebui creată o vedere.



Pasul 2. Indicăm procesorul de texte pentru editarea scrisorii. Cel implicit este Microsoft Word.



Pasul 3. Selectăm opțiunea de creare a documentului. Pentru a selecta un document existent, introducem numele acestuia sau îl căutăm.



Pasul 4. Alegem Form Letter ca tip de document. Celelalte tipuri sunt Label, Envelope sau Catalog.



Pasul 5. Deschidem Word-ul ca editor de texte. Compunem textul scrisorii, având grijă să inserăm câmpul nume din baza de date prin lista Insert Merge Field. Butoanele barei de instrumente care apare aparțin Word-ului și se presupun a fi cunoscute.

Trimitem scrisoarea prin clic pe butonul **Mail Merge** de pe bara de instrumente. Fereastra deschisă de declanșatorul Setup este prezentată în figura 15-6.

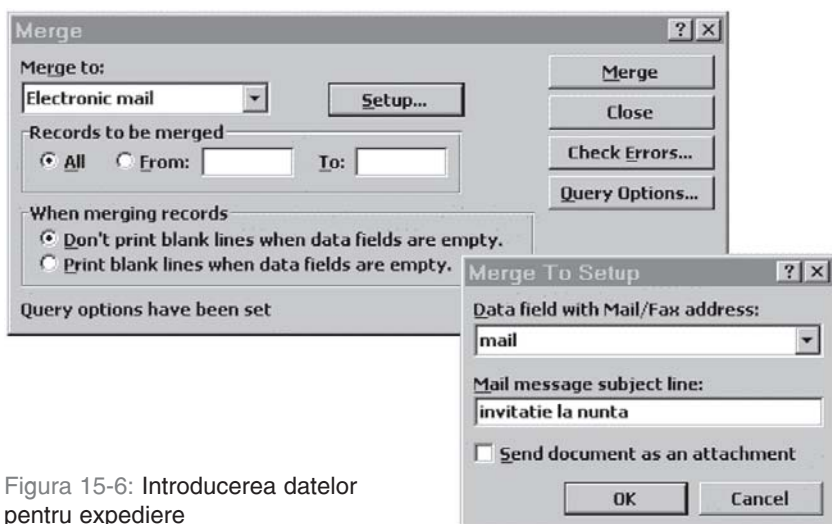


Figura 15-6: Introducerea datelor pentru expediere

Indicăm numele câmpului din tabelă care conține adresa de e-mail a fiecărei persoane. În vederea filtrării datelor (nu toți prietenii vor fi invitați) se poate folosi domeniul de înregistrări sau o condiție pusă prin Query Options. Executăm clic pe butonul Merge pentru a expedia la toate adresele găsite în articolele selectate.



sarcini de laborator

Creați o aplicație în FoxPro pentru concursul **Info-Baby** care se va desfășura în școala dumneavoastră. Se vor afișa pe internet:

- Informațiile generale despre concurs și organizatori, condițiile de participare;
- Programul concursului pe zile, ore, activități, locuri de desfășurare;
- Membrii juriului;
- Concurenții, în ordinea înscrierilor, alfabetic, pe zile;
- Problemele date în fiecare zi;
- Situațiile zilnice – rezultatele obținute de fiecare participant în ziua respectivă, eventual clasamentul până în momentul curent;
- Clasamentul final pe județe, pe grupe de vârstă;
- „Perlele“, evenimentele deosebite, comentariile prietenilor.

RECAPITULARE

Prelucrarea procedurală a datelor în FoxPro și SQL

SINTEZE. Limbajul de programare FoxPro și Limbajul de comenzi SQL

Definirea structurilor de control	<code>IF ... ENDIF DO WHILE...ENDDO SCAN...ENDSCAN FOR ..ENDFOR DO CASE...ENDCASE</code>
Definirea procedurilor și a funcțiilor utilizator	<code>PROCEDURE..ENDPROC FUNCTION...ENDFUNC.. RETURN</code>
Comunicarea între unitățile funcționale	<code>PUBLIC / PRIVATE/LOCAL PARAMETERS SET UDFPARMS</code>
Definirea unei interogări	<code>SELECT...FROM...TOFILE / INTO... ...WHILE...GROUP BY [HAVING]..</code>
Definirea unei tabele	<code>CREATE TABLE (<structura>)... CHECK..FOREIGN KEY..TAG.. REFERENCES...[TAG..]</code>
Utilizarea bazei de date	<code>CREATE / DISPLAY DATABASE OPEN/ CLOSE/ DELETE DATABASE SET DATABASE TO</code>
Utilizarea tabelor într-o bază de date	<code>ADD/ REMOVE/ DISPLAY TABLES</code>
Proceduri stocate	<code>MODIFY/APPEND/DISPLAY PROCEDURE COPY PROCEDURE TO..</code>
Declanșatoare	<code>CREATE/DELETE TRIGGER ONFORDELETE/INSERT/UODATE</code>
Actualizare (sql)	<code>UPDATE SET...WHERE... INSERT...INTO...VALUE... DELETE FROM...WHERE...</code>
Modificarea structurii	<code>ALTER TABLE...</code>

1. Care este diferența între limbajul FoxPro și SQL?
2. Ce alte limbaje relaționale mai există?
3. Cum se asigură comunicarea datelor între unitățile funcționale ale unui program?
Ce sunt variabilele publice?
4. Ce moduri de transmisie a parametrilor cunoașteți? Cum se transmit parametri tip referință?
5. Ce este o interogare? Cum se pot proiecta interogări de totalizare la nivelul unui grup? Dați exemple de funcții agregat!

O stație centrală de pompare trebuie să alimenteze cu apă niște beneficiari, notați cu x_1, x_2, \dots, x_N . Costurile de construcție a conductei de apă între beneficiarul x_i și beneficiarul x_j se cunosc și se dau prin triplete de felul (i, j, k) unde i =codul beneficiarului i , j =codul beneficiarului j , k =costul lucrării de la beneficiarul i la beneficiarul j .

Să se determine schema construcției rețelei de alimentare care să conducă la un cost total de realizare minim.

Indicație:

Problema poate fi rezolvată prin algoritmul lui Kruskall pentru APM, cunoscut din lecțiile de programare din clasa a XI-a.

Punctaj:

1. Crearea structurilor de memorare1p
Se cere ca programul să nu distrugă eventualele date existente; dacă tabela cu toate costurile între beneficiari există, ea trebuie doar deschisă, nu creată. Tabela care va reține ponderile asociate nodurilor este temporară și va fi creată de program.
2. Popularea tabelii cu date (costurile lucrărilor)1p
Se cere ca operația de introducere a informațiilor să fie continuată sau abandonată în funcție de opțiunea operatorului, chiar de la începutul secvenței de citire.
3. Determinarea numărului de beneficiari (noduri)1p
4. Încărcarea cu date a tabelii ponderi1p
5. Ordonarea datelor în funcție de costuri1p
6. Algoritmul Kruskall1p
7. Determinarea sumei costurilor pentru soluția dată1p
8. Afișarea soluției (drumul și suma totală minimă)1p
Se cere afișarea datelor sub forma unui tabel cu titlu, cap-tabel, conținutul pe coloane separate de un caracter („“), o linie care să marcheze terminarea tabelului.
9. Aspect, comentarii, indentarea structurilor1p
10. Oficiu1p

Fie o bază de date necesară evidenței unităților de cazare și turiștilor, cu următoarele tabele:

UNITATI(cod N(5), nume C(15), fel C(1), categorie N(2), agentie C(15)) pentru reținerea unităților de cazare. Numele unității va fi dat cu numele stațiunii pe primele poziții apoi numele unității; separator este caracterul „/”.

Atributul Fel conține o literă: „H” pentru hotel; „M” pentru motel etc.

CAMERE(cod_unit N(4), cod_cam N(3), nr_pat N(1), are_tel L, are_tv L, pret N(5), este_ocup L) pentru informațiile despre camere.

OCUPARE(cod_unit N(4), cod_cam N(3), data_s D, data_p D, nume_pers C(15), b_i C(10)) pentru informații despre turiști.

Cerințe:

- 1) Știind că o nouă unitate de cazare Y are exact aceleași caracteristici cu hotelul „Alfa“ din Sinaia, să se adauge informațiile despre noua unitate în baza de date. (Are aceleași camere, confort etc., dar camerele sunt inițial neocupate!)
- 2) Treceți toate unitățile de cazare din Sinaia în alt fișier, cu structura CAZARE.DBF(cu N(3), sta C(10), nu C(10, fel N(1), cat N(1), ag C(10)), unde cu=codul unității; sta=numele stațiunii; nu=numele unității; fel=tipul unității (astfel: 1=Hotel, 2=Motel, 3=Vila, 4=Căsuța); cat=categoria, ag=agenția de turism)
- 3) Modificați codul unei unități de cazare.
- 4) Pentru toate unitățile de cazare din stațiunea X, aflați numărul de camere neocupate.
- 5) Afișați numărul de camere libere pe unități, sub forma:
Nume-stațiune...unitate...fel...nr-camere...

Notă: Se acordă câte două puncte pentru fiecare sarcină, cu excepția sarcinii 3 notată cu un punct; se acordă un punct din oficiu.

Testul 3

Operații în SQL

Se ține evidența vânzărilor mai multor magazine ale aceleiași societăți comerciale „SC INTIM SRL“ într-o bază de date cu tabelele MAGAZINE (cod-magazin, nume, adresa), PRODUSE (cod-produs, nume-produs, unitate de măsură, preț unitar), RAIOANE (cod-raion, nume), VINZARI (dată, cod-magazin, cod-raion, cod-produs, um, cantitate).

Cerințe:

- A. Să se proiecteze baza de date și tabelele prin comenzi SQL.
- B. Să se scrie comenzile pentru inserarea unei înregistrări noi în fiecare tabelă.
- C. Să se ștergă din tabela VANZARI toate articolele care au data vânzării înainte de 1 ianuarie anul curent.
- D. Magazinul care are numele "Metro-golden" s-a mutat în "str. Iancului, nr. 456". Faceți modificarea corespunzătoare prin comenzi SQL
- E. Să se afle prin comenzi SELECT:
 - cantitatea totală de hârtie vândută azi;
 - numărul total de raioane distincte;
 - valoarea totală a vânzărilor pe magazine;
 - care sunt primele 3 magazine fruntașe (au vândut cele mai multe produse - cantitativ!).

Punctaj:

A. 2p; B., C., D, câte 1 p; E. 4p și 1 p din oficiu.

Elemente de programare orientată spre obiecte

- *Obiecte, proprietăți, evenimente, metode*
- *Comenzi necesare programării obiectuale în Visual FoxPro*

Un obiect este un lucru (o fereastră, un buton radio, un comutator etc.) căruia îi putem defini **proprietățile**, **evenimentele** care pot să acționeze asupra lui și **metodele ca operații asociate sau ca acțiuni de răspuns** la evenimentele care apar.

Să ne amintim câteva obiecte ale interfeței Windows. Principalul element de interfață este fereastra, care poate conține alte obiecte: comutatoare, liste, butoane de comandă etc.

O **proprietate** definește una dintre caracteristicile obiectului ca aspect sau comportament. De exemplu, o fereastră poate fi caracterizată prin: titlu, distanța în pixeli față de marginea din stânga a ecranului, distanța față de marginea superioară a ecranului, înălțimea, starea vizibilă sau ascunsă etc.

Un **eveniment** este o activitate specifică și predeterminată, inițiată de sistemul de operare (eveniment intern) sau de utilizator (eveniment extern), la care un obiect știe să reacționeze. De exemplu un eveniment extern ar fi un clic cu mouse-ul, mișcarea mouse-ului, apăsarea unei taste.

Metodele sunt proceduri asociate unui obiect. Deosebirea dintre metode și evenimente este că primele pot exista și pot fi apelate indiferent de apariția evenimentelor. Evenimentele, de asemenea, pot avea asociate anumite proceduri, acțiuni care vor fi executate ori de câte ori se produce evenimentul respectiv. Metodele sunt asociate și legate cu obiectele cărora le aparțin. De exemplu, o metodă asociată selectării ferestrei poate să-i schimbe culoarea de fond, poziția etc.

Obiectele diferă între ele prin proprietăți. Două declanșatoare pot avea dimensiuni diferite, poziții diferite pe ecran, alt text explicativ etc., dar se comportă similar (răspund la aceleași evenimente prin aceleași acțiuni). Spunem că aparțin aceleiași clase.

O **clasă** este definită prin mulțimea obiectelor care au în comun aceleași proprietăți și același comportament. Prin definirea unei clase specificăm modul în care dorim să se comporte și să arate toate obiectele care îi aparțin.

În limbajul Visual FoxPro au fost deja implementate numeroase **clase de bază predefinite**, pe baza cărora putem crea obiecte ca instanțe ale acestor clase și, desigur, clase derivate. De exemplu, clasele uzuale pentru obiectele de interfață cunoscute sunt: **Form** (pentru formulare sau video-formate); **CheckBox** (pentru casete de validare sau comutatoare); **CommandButton** (pentru butoane de comandă sau declanșatoare); **Listbox** (pentru liste).

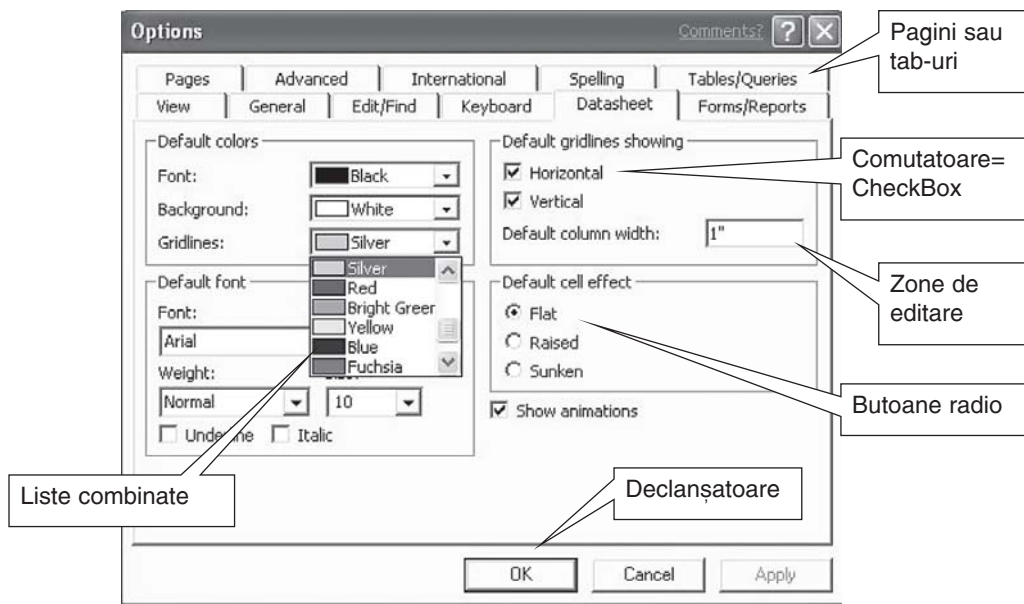


Figura 16-1: Obiecte de interfață cunoscute

Clasele pot fi împărțite în clase **container** și clase **controller**. O clasă **container** poate găzdui alte obiecte aparținând clasei controller sau chiar containere. De exemplu, un formular poate conține zone de editare, ca obiecte din clasa controller, dar și grupuri de butoane (clasa **CommandGroup**, o altă clasă container). Butoanele radio fac parte tot dintr-o clasă container: **OptionGroup**.

Unele clase sunt **non-vizuale**, adică obiectele care le aparțin nu sunt vizibile pe ecran, ci au diferite utilizări, cum ar fi: calculul unor valori, stabilirea momentului de timp pentru a efectua o anumită acțiune etc. De exemplu, clasa **Timer** cronometrează anumite acțiuni și lansează altele în funcție de momente precizate de timp.

Obiectele **vizuale** pot fi:

1. de tip **container** – care conțin alte obiecte; de exemplu un **formular**;
2. de tip **controller** – care nu pot fi părinți; de exemplu un **comutator**.

În Visual FoxPro sunt definite 4 clase de bază de tip container: **CommandGroup**, **Grid**, **OptionGroup** și **PageFrame**.

În plus față de clasele de bază, mai există containere care pot fi conținute numai în alte containere (nu pot exista independent), cum ar fi: **Column** (coloanele unui **Grid** conțin fiecare câte un Header și un control de afișare a datelor, dar nu pot exista în afara grilei care le conține), **Page** (pagina unui **PageSet**, care nu poate exista în afara containerului părinte și poate conține alte obiecte).

Proprietățile definatorii ale claselor

- **Încapsulare** – în definiția unei clase sunt incluse și proprietățile și metodele (vezi comanda **DEFINE CLASS**).
- **Derivare** – este permisă crearea unor subclase sau clase derivate, care vor avea întreaga funcționalitate a clasei părinte, plus alte noi specificații.
- **Moștenire** – proprietățile clasei de bază nu vor mai trebui să fie definite în clasa derivată. Aici se memorează doar proprietățile și metodele noi. Orice schimbare a caracteristicilor clasei de bază este reflectată automat și în clasa derivată.

Datorită acestor caracteristici ale claselor se pot defini **ierarhii de clase** care se memorează în fișiere **biblioteci de clase**. Clasele pot fi introduse, șterse sau modificate într-o bibliotecă. Înainte de utilizarea unei clase existente într-o bibliotecă, este necesară deschiderea bibliotecii.

Vom explicita aceste concepte prin formatele generale ale comenzilor și funcțiilor uzuale de lucru cu obiecte, cu atenționarea că FoxPro pune la dispoziție utilitare rapide pentru construirea și dezvoltarea claselor (de tip Designer, Wizard) pe care le vom prezenta în lecțiile următoare. Deocamdată, vom prezenta comenzile FoxPro pentru definirea și utilizarea claselor și obiectelor.

Comenzile FoxPro necesare programării obiectelor

1. **Definirea** unei clase se face prin comanda:

```
DEFINE CLASS <clasa> AS <parinte> [PROTECTED<lista de
proprietati>] [<nume-proprietate>=<expresie>...]
[ADD OBJECT [PROTECTED] <nume-obiect> AS
<clasa>[WITH <lista-proprietati>]]
[PROCEDURE <nume-procedura><comenzi>ENDPROC]
ENDDFINE
```

Comanda permite crearea unei subclase **<nume-clasa>** din clasa părinte precizată în clauza **AS**. Proprietățile noi ale clasei pot fi protejate – prevenind accesul și schimbarea lor din exterior (clauza **PROTECTED**). Se pot atribui valori proprietăților prin construcții de forma: **<nume-proprietate>=<expresie>**.

Exemple

Exemplul 1: Definirea unei subclase cu numele **Fer1** cu caracteristicile clasei părinte.

```
DEFINE CLASS fer1 AS FORM
ENDDFINE
```

Exemplul 2: O clasă pentru „cărți de vizită” non-vizuală.

```
DEFINE CLASS carti AS CUSTOM
Nume="Popescu"
Ocupatie="artist"
Adresa="Iasi"
ENDDFINE
```

Exemplul 3: Subclasa **fer2** are alte proprietăți: poziție pe ecran, alte dimensiuni, fond galben, text roșu¹.

```
DEFINE CLASS fer2 AS Form
Top=2
Left=10
Hight=10
Width=80
BackColor=RGB(255,255,0)
ForeColor=RGB(128,0,0)
ENDDFINE
```

O definiție de subclasă poate să conțină obiecte ale altor clase. Ele sunt specificate prin clauza **ADD OBJECT**. Clauza **PROTECTED** previne accesul la modificări exterioare ale acestor obiecte. Clauza **WITH** dă proprietățile obiectelor adăugate.

Exemplul 4: Adăugarea unui obiect de tip buton declanșator la o clasă container derivată (**Fer2**). Noua clasă va moșteni textul roșu, poziția (Top=2, Left=10) și dimensiunile (10X80) dar va avea fondul alb.

```
DEFINE CLASS Fer4 AS fer2
Backcolor=RGB(0,0,0)
ADD OBJECT X As CommandButton
WITH Caption='More>>', Top=0, Left=0, Height='4, Widht=12
ENDDFINE
```

Pentru subclasa definită se pot specifica evenimentele și metodele ca set de funcții utilizator sau proceduri.

Evenimentele sunt apelate prin: **<nume-obiect>.<eveniment>**, iar metodele prin construcția: **<nume-obiect>.<metoda>**.

Referințele relative se exprimă prin:

Parent	&& Primul container al obiectului curent
This	&& Obiectul curent
ThisForm	&& Formularul care conține obiectul curent

Exemplul 5. Definim o clasă de butoane (declanșatori) care au scris textul „More>>” și care să coboare cu 40 de pixeli la selectare.

```
DEFINE CLASS buton1 AS CommandButton
Caption="More>>"
PROCEDURE CLICK
Buton1.TOP=50 &&sau This.TOP=50
ENDPROC
ENDDFINE
```

¹ Culorile se definesc indicând cantitatea de Albastru, Galben, Rosu ca valori numerice. Exemplu: RGB(0,0,0)-alb, RGB(255,255,255)-negru, RGB(127,0,0)-galben

Exemplul 6. Definim o fereastră care afișează un mesaj când se execută clic. Clasa de bază este **Form**. Observați moștenirea proprietăților clasei de bază. Definim doar acțiunea la evenimentul clic.

```
DEFINE CLASS fer_mesaj AS Form
  Procedure click
    =MessageBox("in fereastra s-a apasat mouse-ul")
  endproc
ENDDDEFINE
```

2. Crearea unui obiect (instanțiere) se face prin comanda de apel a funcției **CREATEOBJECT**, care întoarce o referință către obiectul creat:

```
<nume-obiect> = CREATEOBJECT (<nume-clasa>)
```

3. Afișarea listei complete a obiectelor active, a proprietăților și valorilor acestora:

```
DISPLAY OBJECT
```

4. Eliberarea obiectelor se face prin ștergerea variabilelor asociate:

```
RELEASE <nume-obiecte>
```

5. Comanda de activare a procesorului de evenimente:

```
READ EVENTS
```

6. Comanda de oprire a procesorului de evenimente:

```
CLEAR EVENTS
```

Exemplul 7. Definim o instanță a ferestrei **Fer1** pe care o afișăm:

```
F1=CREATEOBJECT ("fer1")
F1.SHOW           && observați metoda SHOW de afișare obiect
```

Exemplul 8. Afișarea proprietăților obiectului

```
? F1.Top           && obiectul F1 este o fereastră
? F1.Left          && afișăm poziția colțului dreapta
```

Exemplul 9. Modificăm unele proprietăți odată cu instanțierea:

```
F2=CREATEOBJECT ("fer1")
f2.Caption="alta fereastra"
f2.Show
```

Exemplul 10. Încapsulăm obiecte odată cu definirea unei instanțe a clasei container. Observați metoda **ADDOBJECT** asociată clasei container **Form**; ea se apelează cu **<obiect>.ADDOBJECT (<nume-obiect>, <nume-clasa>)**

```
F3=CREATEOBJECT ("Form")           && clasa Form este container
F3.Show                             && metoda afișare
F3.Caption="fereastra cu buton"    && proprietatea nume-fereastra
F3.ADDOBJECT ("B1", "CommandButtton") && metoda ADDOBJECT
F3.B1.Visible=.T.                  && proprietatea de vizibilitate
F3.B1.Caption='Exit'               && proprietatea nume-buton
```

Exemplul 11. Crearea unor obiecte non-vizuale: în legătură cu o aplicație de gestiune a vânzărilor, definim o clasă denumită *Produce*, care conține atributele *cod*, *nume*, *cant*, *pret* și *valoare*. Atributul *valoare* se calculează printr-o procedură. Instanța cu numele *Birou* atribuie valori atributelor și afișează obiectul.

```
Birou=CREATEOBJECT("Produce")
Birou.Cod='1234'
Birou.Nume=' Birou Student'
Birou.Cant=125
Birou.Pret=1400000
Birou.Calcul_Valoare
DISPLAY OBJECT Birou                                && afișare obiect
DEFINE CLASS Produce AS CUSTOM Store " to cod, nume
Store 0 to Cant, Pret,Valoare
PROCEDURE Calcul_Valoare
THIS.Valoare=THIS.Pret * THIS.Cant
ENDPROC
ENDDFINE
```



sarcini de laborator

I.

1. Creați o clasă cu numele *Fereastra* pentru orice fereastră cu fundal galben, cu proprietățile de a putea fi redimensionată, minimizată, maximizată, mutată, închisă prin butoanele corespunzătoare. Fixați și dimensiunile.
2. Creați două obiecte ale clasei *Fereastra*: un obiect va moșteni culoarea, dar va avea alte dimensiuni, celălalt va moșteni dimensiunile, dar va avea ca fundal o imagine. Numiți-le *Fer1*, *fer2*.
3. Creați un buton de comandă care să se numească "fereastra 1" și care să afișeze prima fereastră la evenimentul clic.
4. Creați un buton de comandă care să se numească "fereastra 2", care să afișeze a doua fereastră la evenimentul dublu clic.
5. Plasați cele două butoane de comandă pe o fereastră cu numele *fer3*.

II. Ce realizează construcțiile următoare?

- a) `Thisform.release`
- b) `Thisform.refresh`
- c) `Thisform.name="frmelevi"`
- d) `Thisform.txtnume.caption="oprescu"`
- e) `This.value=5`
- f) `This.visible=.t.`

Proiectarea formularelor

- *Formulare, machete ecran sau video-formate*
- *Form Designer: componente, mod de utilizare*
- *Proiectarea obiectelor de control*
- *Proiectarea formularelor folosind Form Wizard*

Formularul ca element de interfață cuprinde una sau mai multe ferestre, pe care sunt plasate informații. Formularele pot fi folosite pentru prezentarea aplicației sau drept panou de bord, având butoane care deschid ferestrele specifice anumitor sarcini ale aplicației. Cel mai adesea formularele sunt folosite pentru vizualizarea și editarea datelor din tabelele unei baze de date.

Principalele informații ce trebuie avute în vedere la formatarea ecranului pentru actualizarea datelor pot fi grupate¹ astfel:

1. informații pentru explicarea semnificației câmpurilor din baza de date;
2. câmpurile din baza de date sau variabilele de memorie;
3. mesaje de eroare, date eronate, texte explicative privind corectarea lor;
4. liste de valori posibile care se pot atribui variabilelor;
5. opțiuni utilizator pentru prelucrarea datelor;
6. informații statistice, informații de întreținere și de ghidare a operatorului pe durata sesiunii de lucru;
7. informații interogative și răspunsuri posibile privind continuarea operațiilor sau renunțarea la acestea.

Observați formularul din figura 17-1, realizat cu Form Wizard în FoxPro. Este o fereastră care are proprietățile de a putea fi micșorată, mărită sau închisă prin butoanele situate pe bara de titlu a ferestrei. De asemenea, se poate fixa și titlul ferestrei. Pe fereastră sunt plasate diferite obiecte de control, care permit vizualizarea datelor și editarea lor. Proiectarea formularelor se poate face **scriind cod** (în modul tradițional sau în modul orientat spre obiecte) sau în mod vizual, prin **Form Designer** și **Form Wizard**.

Nr_fact	Data	Cant	Pret
1000	02/05/02	10	100000
1008	02/06/02	35	100000
1011	05/25/02	20	100000
1025	05/26/02	30	100000

Figura 17-1: Exemplu de formular

¹ După I. Lungu, Foxpro pag. 311.

Generatorul de formulare (Form Designer)

Visual FoxPro oferă mai multe posibilități pentru crearea formularelor: **Form Designer**, **Form Builder**, **Form Wizard**.

Form Wizard este apelat selectând **Tools, Wizard** și permite generarea unui program de introducere pe baza unei machete construite automat din informațiile date de operator.

Form Builder permite, de asemenea, o proiectare rapidă, dar pe baza opțiunii Quick Form din meniul Form. Form Designer este folosit pentru aplicații mai complicate, unde cele două instrumente nu fac față.

Form Designer este constructorul de formulare care permite proiectarea interactivă a unui formular (pas cu pas) de către utilizator. Deschide o fereastră de proiectare pe care se pot plasa obiectele de control necesare.

Apelarea utilitarului **Form Designer** se face selectând **File, New, Form** sau prin comanda

```
CREATE FORM <fis.Scx>
```

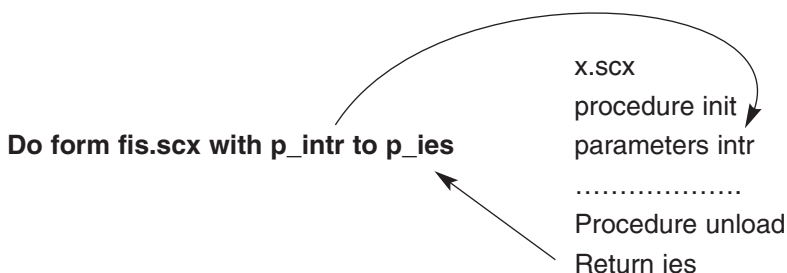
După proiectarea sa, un formular este memorat într-un fișier cu extensia **.scx** care poate fi deschis cu **USE** și vizualizat cu **BROWSE**.

Lansarea formularului se face prin comanda:

```
DO FORM <fis.scx> [WITH <param-intr>] [TO <param-ies>]
```

Comunicarea formularului cu programul apelant se poate face prin parametri. Parametrii de intrare se pun în clauza **With** iar parametrul de ieșire (atenție, este unul singur, o variabilă) în clauza **TO** a comenzii **DO**.

Dacă se folosesc parametri, macheta trebuie definită cu proprietatea **WindowType=1** și trebuie să aibă metoda **Init** cu instrucțiunea **Parameters <lista-var>** pentru intrări. Returnarea unui rezultat se face prin comanda **Return <exp>** pusă în metoda **Unload**.



Mediul de lucru

Cuprinde o fereastră de proiectare pe care vor fi dispuse obiectele de control necesare, fereastra de proprietăți, ferestre de dialog, meniuri și bare cu butoane.

Bara de instrumente Form Designer

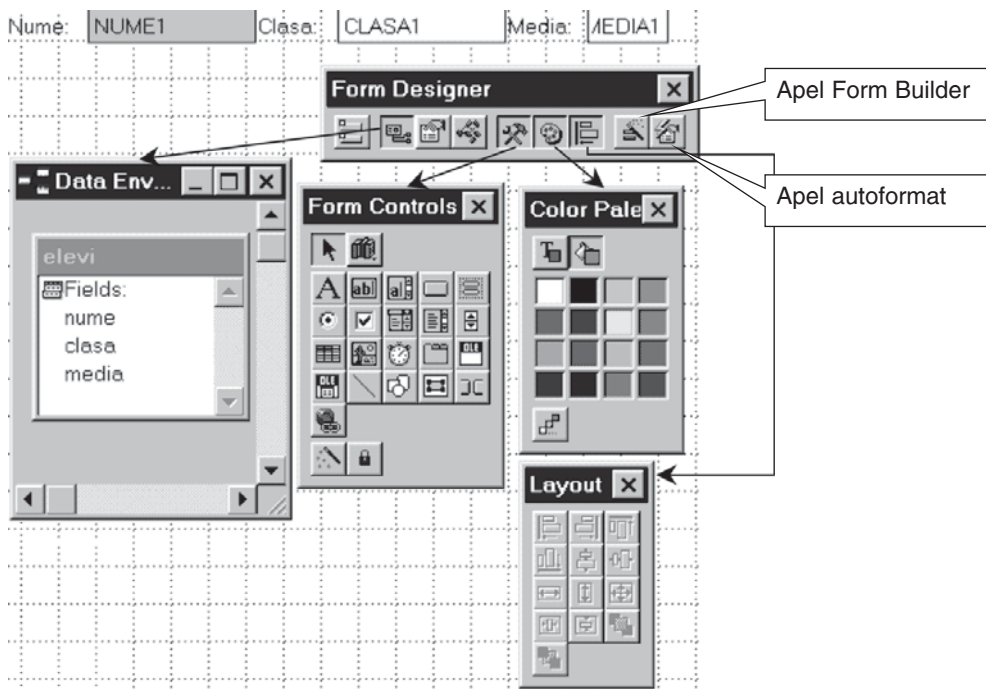


Figura 17-2: Bara de instrumente Form Designer

Fereastra Properties

Caracteristicile fiecărui obiect plasat pe formular pot fi observate în fereastra **Properties**, o fereastră comună tuturor obiectelor. Aceasta permite atât vizualizarea proprietăților, cât și introducerea codului pentru metode și evenimente. Se poate deshide selectând **View, Properties** din meniul contextual asociat ferestrei Form Design sau cel asociat obiectului pe care dorim să-l edităm.

Fereastra Properties afișează pe prima linie numele obiectului curent, ale cărui proprietăți sunt afișate. Deschizând lista putem să observăm ierarhia obiectelor din formularul curent.

Fereastra are mai multe tab-uri (All, Data, Method, Layout, Other) și o zonă de afișare cu două coloane: prima pentru numele caracteristicii, iar a doua pentru valoarea sa. Iată pe scurt conținutul tab-urilor: **ALL** afișează toate proprietățile, **Data** afișează proprietățile legate de date, **Method** afișează metodele, **Layout** afișează proprietăți legate de aspect (culoare, font etc.), **Other** afișează alte proprietăți.

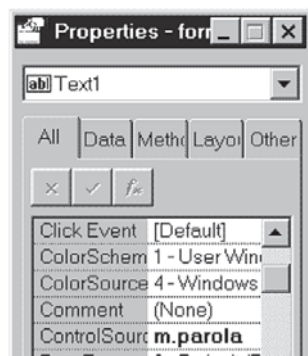


Figura 17-3: Fereastra Properties

Meniul Form

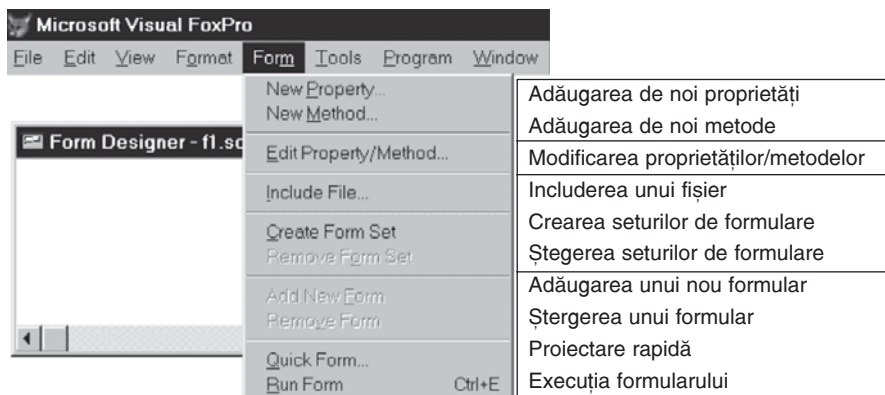


Figura 17-4: Meniul Form

Obiectul Data Environment

Utilitarul Form Designer, ca și alte utilitare, permite în timpul activității de proiectare deschiderea tabelor, specificarea indecșilor și a legăturilor. Aceste informații sunt depuse în obiectul **Data Environment** fără ca utilizatorul să aibă cunoștință de acest lucru. La lansarea formularului, întreaga activitate de deschidere a tabelor și de restabilire a legăturilor este realizată automat. Închiderea formularului înseamnă și închiderea tabelor folosite.

Deci nu trebuie scrisă nici o comandă de deschidere sau închidere! Dacă, totuși, dorim acest lucru, metoda **Load** este potrivită pentru comenzile de deschidere, iar metoda **Unload** pentru cele de închidere.

Putem deschide ecranul obiectului Data Environment prin meniul contextual atașat formularului (clic dreapta oriunde pe fereastra de proiectare), prin meniul Form sau prin bara de instrumente Form Designer.

Pentru deschiderea fișierelor necesare proiectului este folosit **meniul contextual** al obiectului (clic dreapta pe obiectul Data Environment, apoi selectăm **Add**). Pentru ștergerea unei tabele folosim **Remove**, iar pentru vizualizarea conținutului, **Browse**. Adăugarea unei noi tabele presupune și specificarea legăturilor. Dacă tabele fac parte dintr-o bază de date, utilitarul Form Designer folosește legătura permanentă existentă. Putem păstra această legătură, o putem schimba sau putem plasa una nouă.

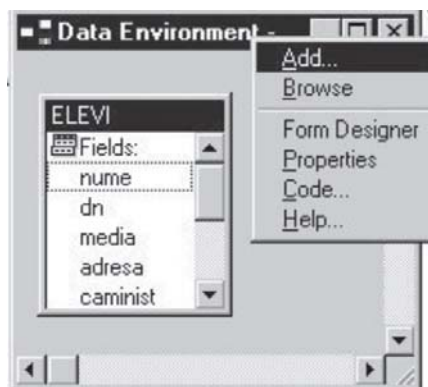


Figura 17-5: Obiectul Data Environment

Meniul View

View	Format	Form	Tools
Tab Order			Schimbarea ordinii obiectelor de pe formular
Data Environment...			Deschiderea ferestrei pentru mediul de date
✓ Properties			Deschiderea ferestrei Properties
Code			Deschiderea ferestrelor de proceduri
✓ Form Controls Toolbar			Afișarea barei de instrumente Form Controls
✓ Layout Toolbar			Afișarea barei de instrumente Layout
✓ Color Palette Toolbar			Afișarea barei de instrumente Color Palette
✓ Grid Lines			Afișarea/ascunderea liniilor de grilă
Show Position			Vizualizarea poziției obiectului în proiectare
Toolbars...			Alegerea barelor de instrumente ce vor fi afișate

Figura 17-6: Meniul View

Proprietăți și metode specifice unui formular

- **WindowType** – specifică modul de rulare a unui formular. Valoarea zero specifică un formular *normal*, care poate preda controlul altor ferestre deschise simultan pe ecran; valoarea 1 arată un formular *modal*, care nu permite trecerea controlului la alt formular. Forma modală trebuie să fie folosită atunci când, de exemplu, există un program care transmite un parametru formularului, îl deschide și așteaptă un răspuns la terminarea lucrului cu acesta. Comunicarea programului cu formularul se face prin parametri, dar numai dacă fereastra este de tip modal.
- **Closable, Movable, MaxButton, MinButton, ControlBox** pe valoarea .T. dau ferestrei proprietățile de a putea fi închisă, mutată, de a avea butoanele de maximizare, minimizare și meniul standard în colțul din stânga sus.
- **Load** – este o metodă apelată imediat după crearea formularului și poate conține comenzi de deschidere a unor tabele sau de inițializare a unor variabile.
- **Init** – este o metodă apelată la crearea formularului (la rulare). Ea este cea care preia parametrii de rulare ai formularului, trimiși de către programul apelant. Prima linie a acestei proceduri conține comanda **Parameters**.
- **Destroy** – este o metodă apelată la eliminarea formularului de pe ecran.
- **Unload** – este o metodă apelată la distrugerea obiectului din memorie; este ultima metodă.

Obiectele de control sau de interfață

Obiectele de control pe care le poate conține un formular pot fi specificate prin bara de instrumente **Form controls**.

Dintre butoanele pe care le vom folosi în lecțiile următoare pentru proiectarea obiectelor de control, le explicăm pe ultimele două:

- ⇒ **BuilderLock** – comută în modul de generare, apelând sau ieșind din utilitarul Builder. Putem proiecta manual fiecare obiect de control, plasându-l pe suprafața de lucru, conturând dimensiunea și setându-i proprietățile în fereastra Properties. O altă modalitate este să folosim un asistent numit Builder, care

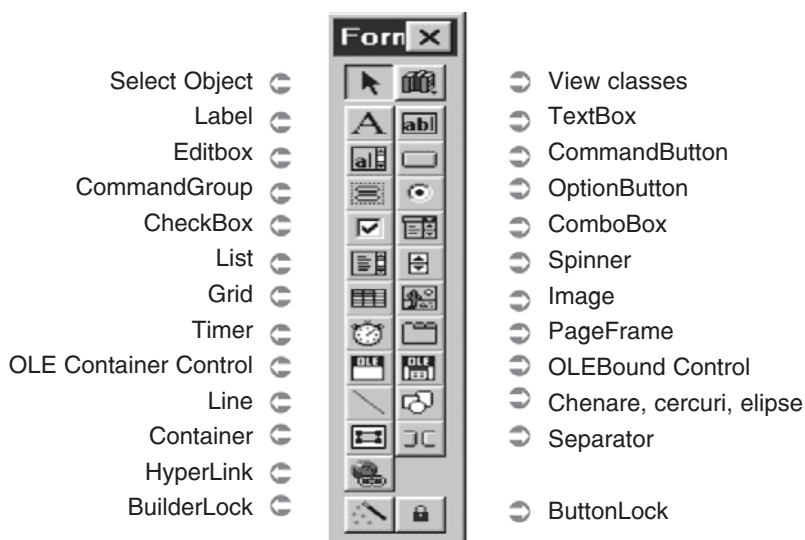


Figura 17-7: Bara de instrumente Form Controls

prezintă ecrane de dialog pentru construirea rapidă a obiectului. (Nu este posibil chiar pentru toate obiectele, dar pentru majoritatea se poate și vom prezenta modalitatea de lucru în temele următoare.)

- **ButtonLock** – permite adăugarea de instanțe multiple ale aceluiași obiect. Manevra obișnuită pentru crearea unui obiect este selectarea butonului corespunzător de pe bara de instrumente FormControls, apoi tragere și plasare pe suprafața de lucru. Dacă dorim plasarea aceluiași obiect, se poate folosi ButtonLock. Selectați-l, apoi selectați obiectul dorit (un CheckBox, de exemplu). Executați tragere și plasare pentru o instanță, clic pentru a doua etc.

Operații generale cu obiectele de control

- *Selectarea obiectului* se face prin clic pe suprafața obiectului.
- *Selectarea mai multor obiecte* se face prin butonul de selectare de pe bara de instrumente, apoi tragere și plasare pe suprafața tuturor obiectelor pe care dorim să le selectăm.
- *Mutarea obiectului (sau a grupului)* se face prin tragere și plasare în noua poziție sau prin Clipboard, cu operațiile **Cut**, apoi **Edit**, **Paste**.
- *Redimensionarea* unui obiect selectat se face prin poziționarea cursorului pe o margine a obiectului până apare săgeata dublă, apoi tragere și plasare până la dimensiunile dorite.
- *Copierea unui obiect* se realizează prin intermediul Clipboard-ului, prin operațiile **Copy**, apoi **Edit**, **Paste**. În mod implicit, copia are aceleași proprietăți și metode ca și originalul.

- Ștergerea unui obiect selectat se face apăsând tasta **Delete**.
- Aranjarea obiectelor pe formular se face după dorința utilizatorului. Pentru aranjarea mai ușoară, suprafața formularului este împărțită în pătrate (care pot fi ascunse prin debifarea opțiunii **Grid lines** din **Forms, Options**).
- Stabilirea ordinii obiectelor se face cu **Tab Order**. În mod implicit, ordinea de parcurgere a obiectelor este ordinea creării lor. După apăsarea butonului **Tab Order**, fiecare obiect va avea atașat un număr; schimbarea acestuia se face prin clic pe obiecte în ordinea dorită.
- Schimbarea fontului și a culorii se face pentru obiectele care afișează text prin proprietățile **FontName** și **FontSize**.
- De asemenea, se pot seta caracteristicile de îngroșare (**FontBold**), înclinare (**FontItalic**), subliniere (**FontUnderline**).
- Pentru culoarea textului este folosită proprietatea **ForeColor**, iar pentru specificarea culorii pe care o va avea obiectul atunci când va fi inactiv se folosește proprietatea **DisabledForeColor**.

Proprietăți și metode generale ale obiectelor de interfață

- **Name** – specifică numele obiectului;
- **Top, Left, Width, Height** – specifică poziția și dimensiunea;
- **Caption** – dă titlul obiectului;
- **AutoCenter** – stabilește plasarea automată în centrul ecranului;
- **Show** – stabilește afișarea sa pe ecran;
- **Hide** – stabilește ascunderea obiectului;
- **Activate (metodă)** – stabilește activarea obiectului la poziționarea cursorului pe obiect;
- **Deactivate (metodă)** – stabilește când se predă controlul asupra altui obiect;
- **GotFocus** – stabilește evenimentul prin care obiectul primește controlul, devine activ;
- **LostFocus** – este evenimentul de pierdere a controlului de către obiect;
- **SetFocus** – este evenimentul de fixare a controlului pe obiect.

Proiectarea unui formular prin Quick Form

Un formular simplu pe care îl putem apoi folosi și/sau dezvolta poate fi creat rapid în Visual FoxPro. Este folosit utilitarul **Form Designer**, care poate fi lansat prin **File, New, Form**, iar din meniul **Form** se alege **Quick Form**. Să urmărim modul de lucru prin rezolvarea unei probleme.

Tema 1. Proiectarea rapidă a unui formular prin Quick Form

Problema

Fișierul Elevi conține câmpurile nume C(10), media N(5,2), clasa C(3). Să se realizeze un formular care permite adăugarea unui nou elev.

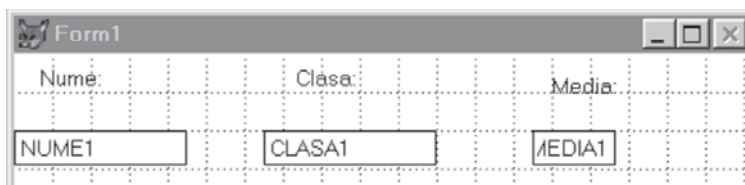


Figura 17-8: Formularul pe care dorim să-l creăm

- Pasul 1) Deschidem fișierul Elevi;
- Pasul 2) Lansăm generatorul de formulare prin CREATE FORM sau prin **File, New, Form, New File** (vom lansa utilitarul Form Wizard într-o lecție viitoare);
- Pasul 3) Alegem opțiunea **Quick Form** din meniul **Form**. Se deschide fereastra **Form Builder**, care are 2 tab-uri:


1. **Field selection** – permite selectarea tabelului și a câmpurilor care se vor citi.
2. **Style** – permite fixarea stilului de formular, la fel ca la Form Wizard.




Figura 17-9: Fereastra Form Builder

- Pasul 4) Selectăm toate câmpurile, fixăm aranjarea pe coloane și ieșim din Form Builder. Vom deplasa cu mouse-ul obiectele, astfel încât numele câmpului să se afle deasupra zonei de editare.
- Pasul 5) Pentru că formularul este folosit la adăugarea unui singur elev, plasăm comanda **APPEND BLANK** în metoda **INIT** a formularului.

În fereastra Properties, ne poziționăm pe prima linie la nivelul obiectului Form1 (observați numele atribuit obiectelor de către generator!), iar în tab-ul **Method** alegem **Init**. Se deschide fereastra de coduri și scriem comanda Append Blank.

- Pasul 6) Lansăm în execuție, de probă, prin butonul **Run:** .
- Pasul 7) Introducem date și deschidem fereastra Browse pentru verificare.
- Pasul 8) Salvăm.

În concluzie: Cum se lucrează cu un formular ?

1. Prima etapă este apelarea utilitarului de proiectare.
2. Definim proprietățile și metodele formularului.
3. Deschidem baza de date.
4. Includem obiectele de control și realizăm un prototip al formularului.
5. Legăm aceste obiecte cu câmpurile din baza de date sau variabilele unde vor fi citite/editate valorile.
6. Verificăm funcționarea formularului prin butonul sau comanda Run. 
7. Putem reveni în proiectare prin clic pe butonul Modify.
8. Salvăm prin **File, Save**.
9. Lansăm în execuție prin comanda **Do Form**.
10. Formularul poate primi valori din exterior (ca parametri de intrare) și/sau poate comunica valori obținute prin acțiunea operatorului (ca parametri de ieșire).



sarcini de laborator


Exerciții cu Quick Form și primele manevre cu un formular:

1. Proiectați un formular de culegere a datelor, utilizând opțiunea Quick Form pentru fișierul ELEVI. Aranjați câmpurile pe linii.
2. Vizualizați caracteristicile tuturor obiectelor din formular și răspundeți la întrebările:
 - a) Ce valoare s-a atribuit proprietății ControlSource la nivelul formularului? Dar pentru primul câmp Nume?
 - b) Care este culoarea de fundal a formularului? Puteți să o schimbați cu galben? Cum sunt codificate culorile?
3. Fixați posibilitatea de adăugare a unui nou articol dacă treceți (cu bine) de ultimul câmp!
4. Ce manevră trebuie să faceți pentru a mări tabela ELEVI prin adăugarea unui ultim articol vid la ieșirea din formular?
5. Scrieți programul care apelează formularul de culegere. Schimbați dimensiunea formularului, centrați-l!
6. Ce manevră trebuie să faceți pentru a șterge din tabelă ultimul articol vid, adăugat fictiv la ieșirea din formular?
7. Închideți fișierul Elevi în fereastra de comenzi și executați o machetă anterior definită, prin comanda **DO FORM**. Ce se observă? Aveam fișierul Elevi închis, și totuși formularul permite citirea câmpurilor. De ce?
8. Introduceți valori și închideți formularul prin clic pe butonul de închidere.
9. Încercați să vedeți efectul în Browse. Ce se observă? Fișierul Elevi este închis sau deschis? Cum vă explicați?
10. Schimbați culoarea de fundal a formularului în albastru deschis. Puneți bordură roșie. Aranjați formularul în centrul ecranului.
11. Încercați să vedeți conținutul fișierului ELEVI după fiecare rulare a programului sau chiar în momentul încărcării formularului.
12. Construiți formulare de culegere a datelor pentru tabelele bazei de date DESFACERE.

Proiectarea vizuală a obiectelor de interfață

Obiectele de control sunt așezate pe suprafața de lucru a formularului, ordinea lor de activare fiind cea de la creare. Se poate schimba ordinea la proiectare selectând **View, Tab Order** sau prin clic pe butonul Tab Order de pe bara de butoane Form Designer. Când se deschide formularul, ținta este fixată, implicit, pe primul obiect. Trecerea de la un obiect la altul se face prin tasta Tab sau cu mouse-ul.

a) Mesaje. Obiecte Label

Obiectul de tip mesaj se proiectează prin butonul Label . Proprietățile uzuale, ca Top, Left, Width, Height se stabilesc automat la plasarea obiectului și aranjarea sa în fereastră. Textul explicativ este valoarea proprietății **Caption**. Alinierea textului în cadrul obiectului se face prin proprietatea **alignment**, care poate lua valorile: 0=la stânga (implicit), 1=la dreapta, 2=centrat. Pentru text sunt utile proprietățile: **FontName, FontSize, FontBold, FontItalic, FontUnderline** etc. Culoarea cernelii este dată de **ForeColor** și cea a fundalului de **BackColor**. Proprietatea **WordWrap=.T.** permite trecerea textului pe mai multe linii dacă lungimea sa nu încapă în dimensiunea orizontală (Width) a obiectului.

O proprietate interesantă este **Visible**, care dă posibilitatea ascunderii unui mesaj sau plasării în același loc a unor mesaje diferite și a controlării momentului în care va fi vizibil fiecare.

Fie următorul mesaj, afișat într-o fereastră. Realizarea lui presupune includerea unui obiect de tip Label și atribuirea proprietăților:

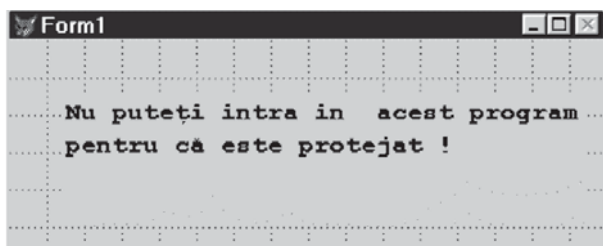


Figura 17-10: Exemplu de mesaj

FontBold=.T., FontName="Courier-R", FontSize=12, WordWrap=.T., Caption= „Nu puteți intra în acest program pentru că este protejat !“.

b) Texte explicative. Obiecte TextBox

Definirea unui obiect de acest tip se face prin butonul TextBox de pe bara de instrumente. Câmpul de editare trebuie să permită citirea și/sau modificarea fie a unei variabile, fie a unui câmp dintr-o tabelă de date, care este specificată în proprietatea **ControlSource**.

Dacă zona va trebui să afișeze numai valoarea, nu să o editeze, vom seta parametrul **ReadOnly=.T.**

- Dacă un câmp de editare nu trebuie să fie accesibil, folosim proprietatea **Enabled=.T.**
- altă proprietate interesantă este **SelectOnEntry**. Când are valoarea .T., va fi selectată automat valoarea din câmpul de editare atunci când acesta devine ținta intrărilor. O simplă apăsare pe tastă și conținutul selectat va fi înlocuit cu tasta respectivă.
- Pentru parole folosim proprietatea **PasswordChar="x"**, specificând caracterul de substituție la introducerea textului.

Metode specifice

- **Valid** – este folosită atunci când terminăm introducerea unei valori în câmp. Dacă valoarea îndeplinește condiția de validare, controlul poate trece la alt obiect. Dacă nu, revenim la editarea valorii.
- **When** – se folosește înainte ca obiectul să devină ținta intrărilor. Valoarea logică a acestei proceduri determină accesul la câmp sau interzice acest lucru.
- **KeyPress** – este o activitate lansată atunci când se apasă o tastă. Procedura începe cu instrucțiunea **Parameters x,y**, unde **x** conține codul tastei apăsate, iar **y** un număr care indică dacă a fost apăsată una dintre tastele de control (Shift, Ctrl, Alt).
- **Interactivechange** – este o activitate de schimbare imediată a valorii unui alt obiect chiar în timpul introducerii datelor în obiectul curent.

Editarea rapidă. Folosirea asistentului TextBox Builder

Pentru a intra în programul TextBox Builder afișăm meniul contextual al câmpului de editare, din care alegem **Builder**.

Utilitarul apare ca o fereastră cu 3 tab-uri, în care răspundem la întrebări fără să fim nevoiți să memorăm numele atributelor și metodelor. Prima întrebare la care răspundem este ce tip de valori va edita câmpul; în funcție de acesta, vor fi schimbate și CheckBox-urile (casetele de validare), fiind adaptate la tipul datei. În tab-ul **Style** setăm aspectul dorit pentru obiect (bi- sau tridimensional, cu bordură, modul de aliniere etc.).

Size text box to fit permite modificarea automată a casetei de editare în funcție de mărimea câmpului. Tab-ul al treilea, **Value**, permite alegerea tabelului/view și a câmpului. Corespunde clauzei **ControlSource**.

Exemple

Exemplul 1. Să realizăm un formular care citește o parolă și o întoarce programului apelant.

Obiectul folosit este de tip **TextBox**, iar după plasarea sa pe formular vom seta proprietățile (vezi prima coloană a tabelului de mai jos). Va trebui însă ca la ieșirea din formular șirul introdus drept parolă în TextBox să fie comunicat programului apelant. Pentru aceasta vom lansa formularul prin comanda **Do With TO**, iar pentru obiectul Form vom scrie metodele **Init**, **Load**, **Unload**.



Figura 17-11: Formularul pentru introducerea parolei

Text1 (parola)	Metoda Init	Metoda Load	Metoda Unload
Passwordchar="X"	Parameters	parola=	Return
Controlsource=m.parola	parola	"initial"	parola

Pași:

1. salvăm cu numele **x.scx** și închidem generatorul.
2. scriem codul de apel al formularului în **cuvant.prg**

```
parola='  
do form x with parola to parola  
? parola  
return
```


Exemplul 2. Să realizăm un formular pentru completarea taxei de cămin. Presupunem că fișierul Elevi este completat deja. Formularul va afișa doar numele fiecărui elev (nu este permisă modificarea lui), va permite completarea câmpului căminist și, dacă este nevoie, a câmpului taxa. Desigur, parcurgerea tabelii Elevi se va face până la ultimul articol dacă variabila pe care o folosim pentru decizia continuării machetei ia valoarea 'D'.

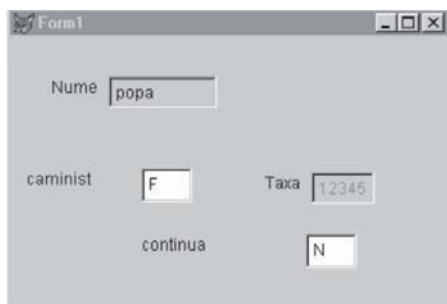


Figura 17-12: Formularul pentru taxa de cămin

Pași:

1. Deschidem utilitarul Form Designer și adăugăm în Data Environment tabela Elevi.
2. Plasăm obiectele pe ecran (Label și TextBox). Observați principalele proprietăți și metode în tabelul următor.

<pre>form1 metoda Init variabila='n'</pre>	<pre>Text2 (caminist) controlSource=caminist metoda Valid if this.value=.t. thisform.text3.enabled=.t. else thisform.text3.enabled=.f. endif</pre>	<pre>Text4 (variabila – continuare) Value = 'n' ControlSource= "m.variabila" metoda Valid if this.value`'dD' if eof() go bottom wait window 'sunteti deja la final' else skip endif endif</pre>
<pre>Text1 (nume) ControlSource ="elevi.nume" ReadOnly= .T.</pre>	<pre>Text3 (taxa) Enabled=.t. ControlSource="elevi.taxa"</pre>	

3. Executăm clic pe butonul **Run** și observăm efectul în fereastra Browse.
4. Salvăm formularul sub numele **adaug.scx**; o executăm prin **Do adaug.scx**.

c) Zone de editare. Obiecte EditBox

Obiectele de tip **EditBox** permit introducerea textelor lungi într-o fereastră de editare și lucrul cu acest text asemeni oricărui editor de texte (de gen Notepad). De obicei, sunt folosite pentru câmpurile Memo din tabele, dar se pot edita chiar fișiere de tip text. Important este că obiectul permite aranjarea automată a textului și mutarea cursorului în text folosind tastele săgeți, PgUp, PgDn, precum și barele de rulare. Sunt disponibile comenzile meniului Edit: Cut, Copy, Paste și combinațiile de taste cunoscute: Ctrl+c=copiere, Ctrl+x=tăiere, Ctrl+v=inserare. Alte proprietăți specifice sunt:

- **AllowTabs** – este permisă introducerea caracterului Tab în text (altfel, știm că tab este folosit pentru părăsirea obiectului și trecerea la următorul obiect);
- **ReadOnly** – permite doar vizualizarea textului.

Editarea rapidă a zonelor de text cu Edit Box Builder

Pentru crearea rapidă a unui obiect zonă de editare se poate folosi utilitarul Edit Box Builder, care poate fi lansat din meniul contextual atașat obiectului după dispunerea acestuia pe formular. Ca și la Text Box Builder, vom răspunde la întrebări grupate pe 3 tab-uri:

1. Format – pentru opțiunile de formatare a a textului.
2. Style – pentru efectele vizuale ale obiectului (Special effect), chenare, alinierea textului.
3. Value – unde va fi depus textul după editare, într-un câmp al unei tabele sau într-o vedere.

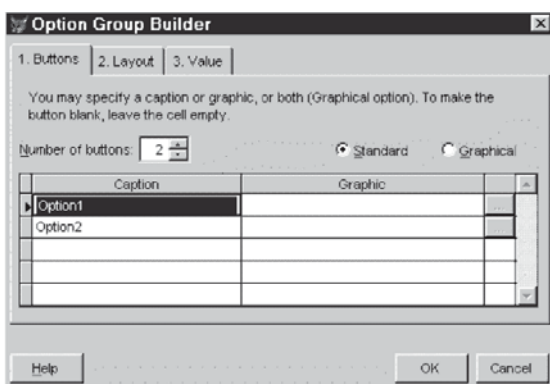


Figura 17-13: Zonă de editare

Exemplu

Fie tabela Elevi în care câmpul Adresa este de tip Memo. Dorim un formular pentru introducerea datelor generale despre un nou elev, inclusiv a adresei sale, folosind obiectul EditBox. În câmpul adresa vor fi scrise pe rânduri separate localitatea, strada și telefonul elevului.

Pași:

1. Deschidem Data Environment cu tabela ELEVI.
2. Plasăm câmpurile pe formular, precedându-le de texte explicative.
3. Pentru fiecare indicăm sursa de date.

TextBox

```
ControlSource= "Elevi.nume"
```

EditBox

```
ControlSource="elevi.adresa",  
AllowTabs=.t.  
ScrollBars=2  
Alignment=1.
```

Atenție! O altă posibilitate de plasare rapidă a câmpurilor pe suprafața formularului este următoarea: deschideți ecranul Data Environment, apoi trageți și plasați de la câmpul dorit pe suprafața de lucru.

Observați asocierea implicită a obiectelor de tip Label cu numele câmpului, a celor de tip TextBox cu câmpurile caracter, a celor EditBox cu câmpurile Memo și a celor de tip CheckBox cu câmpurile logice. Se pot fixa și alte clase pentru câmpuri (de exemplu Spinner pentru câmp Numeric). Asocierea se face direct în proiectarea structurii prin Table Designer.

d) Comutatoare. Obiecte CheckBox

Comutatoarele sau casetele de validare se pot proiecta prin butonul Checkbox. Dintre proprietăți amintim:

- **AutoSize=.T.** permite autodimensionarea obiectului în funcție de textul introdus ca valoare a proprietății.
- **Caption** reprezintă textul explicativ.
- **Enabled=.T.** permite activarea obiectului
- **ControlSource** este numele variabilei sau al câmpului unde se va realiza citirea.

Metoda folosită este **Click**, care va trebui să precizeze acțiunea la apăsarea butonului mouse-ului.

Exemplu

Dezvoltăm formularul pentru adăugarea elevilor, introducând și informația din câmpul de tip logic Căminist. În plus, pentru continuarea introducerii datelor despre un nou elev, am folosit tot un comutator.

1. Deschidem proiectul anterior cu **Modify Form**.
2. Deschidem Data Environment și adăugăm tabela Elevi.
3. Plasăm obiectele pe suprafața de lucru, selectându-le din Form Controls și le fixăm proprietățile și metodele:



The screenshot shows a form with the following elements: a text box labeled 'Nume' with a value of '///', a text box labeled 'Media' with a value of '///', and a checkbox labeled 'Caminist' which is checked. Below the 'Caminist' checkbox is another checkbox labeled 'continuali' which is unchecked.

```
Check2
Caption="continuti?"
ControlSource=
m.continuare
Metoda Valid
If m.continuare
Append blank
Endif
Check1
Caption="Caminist "
ControlSource=
"elevi.caminist"
```

```
Form1
Metoda init
Continuare=.F.
Append blank
```

e) Declanșatoare. Obiecte CommandButton și CommandGroup

Declanșatoarele sunt obiecte simple de tip CommandButton. Proprietățile mai importante sunt:

- **Caption** reprezintă textul afișat pe buton;
- **WordWrap=.T.** permite continuarea textului pe linia următoare; **WordWrap=.F.** trunchiază textul după în Caption la dimensiunea zonei;
- **AutoSize=.T.** permite autodimensionarea butonului în funcție de lungimea textului;
- **Picture** permite afișarea unui fișier imagine (.bmp) pe buton când este neapăsat, **DisablePicture** permite afișarea imaginii când butonul este dezactivat, **DownPicture** permite afișarea unei imagini când butonul este apăsat;

- **Cancel=.t.** arată că obiectul este declanșator de ieșire implicit la tasta Esc (se va executa codul din click la apăsarea tastei Esc);
- **Default=.t.** arată un declanșator implicit la tasta Enter.

În general, nu se asociază câmpuri unui astfel de obiect, ci doar acțiuni, de aceea metoda principală este **Click**, care înseamnă selectarea butonului și determină execuția codului precizat la metodă. Alte evenimente sunt **DbClick**, **RightClick**, **MiddleClick**, **Rightclick**, care nu mai necesită explicații, numele evenimentului sugerând momentul declanșării lui.

Butoanele de comandă se pot grupa, formând un obiect de tip **container CommandGroup**. Avantajul constă în faptul că se poate asocia un cod comun metodelor o singură dată pentru toate butoanele aparținând grupului. După plasarea obiectului pe suprafața de proiectare se poate edita fiecare componentă prin meniul contextual asociat obiectului container la opțiunea Edit. Putem deci schimba pozițiile, dimensiunile fiecărei componente etc.

Proprietăți ale unui grup de butoane:

- **Value** – numărul butonului care a fost selectat din grup.
- **ButtonCount** – numărul butoanelor.

Editarea rapidă a butoanelor de comandă prin Command Group Builder

Și pentru acest obiect de control există un asistent de editare rapidă, numit Command Group Builder, care poate fi lansat din meniul contextual atașat obiectului, după depunerea lui pe suprafața de lucru. Primul tab permite specificarea numărului de butoane și atribuirea unui text sau a unei imagini. Al doilea tab permite alegerea designului obiectului.

Exemple

Exemplul 1. Adăugăm la formularul de citire a elevilor două butoane: butonul **continua** și butonul **gata**.

La activarea butonului **continua** sunt acceptate datele de pe ecran și se trece la un alt articol. La activarea butonului **gata** se termină execuția formularului și se afișează fereastra Browse pentru tabela activă.



Figura 17-15: Cele două butoane adăugate în formular

Pași:

1. Deschidem cu **Modify Form** proiectul anterior de formular;
2. Adăugăm cele două butoane de control și definim proprietățile și metodele **click** asociate:

Cmd1

```
Caption="continua"  
Procedure Click  
Append blank  
endproc
```

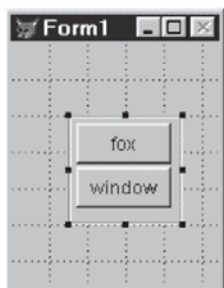
Cmd2

```
Caption="gata"  
Procedure click  
Browse  
endproc
```

Exemplu 2. Proiectăm două butoane de comandă. La apăsarea oricăruia dintre butoane va fi afișat un mesaj cu numele butonului selectat și se va închide formularul.

Pași:

1. Proiectăm grupul și punem la nivelul acestuia proprietățile `ButtonCount=2`, `AutoSize=.t.` și metoda `Click` (vezi procedura asociată).
2. Edităm grupul prin clic dreapta, apoi selectând `Edit`.
3. Pentru primul buton definim proprietatea `Caption='fox'`, iar pentru al doilea `Caption='window'`.



```
Metoda click pentru grup
do case
case this.value=1
wait window 'ati apsat pe butonul'+
  this.command1.caption
thisform.release
case this.value=2
wait window 'ati apsat pe butonul'+
  this.command2.caption
thisform.release
endcase
```

Figura 17-16: Cele două butoane de comandă pe formular

f) Butoane radio. Obiecte `OptionButton` și `OptionGroup`

Butoanele radio sunt folosite pentru selectarea rapidă a unei singure valori dintre cele afișate. `OptionButton` este un obiect care afișează o valoare care poate fi sau nu selectată. Nu apare singur, ci în grup, și formează obiectul container `OptionGroup`. Scopul butoanelor radio este să permită alegerea dintr-o serie de variante a uneia singure. Spre deosebire de un grup de comandă, pe care utilizatorul poate să-l ignore, un grup de butoane radio are întotdeauna un buton selectat.

- **AutoSize=.t.** permite autodimensionarea zonei ocupate pe ecran de buton pentru ca întregul text explicativ să încapă;
- **Caption** reprezintă textul explicativ asociat butonului;
- **ButtonCount** este numărul de butoane din grup;
- **ControlSource** reprezintă variabila sau câmpul unde va fi memorat butonul selectat. Dacă acest câmp este numeric, va fi memorată poziția în grup, iar dacă este șir de caractere, va fi memorat textul explicativ asociat butonului selectat;
- **Value** reprezintă butonul radio care dă valoarea implicită grupului;
- **BorderStyle** asociază un chenar grupului; controlează transparența sa.

Cea mai importantă metodă este **Click**, care determină selectarea obiectului și deselectionarea celorlalte butoane radio din grup.

Editarea rapidă a obiectelor prin OptionGroup Builder

Pentru editarea butoanelor radio există un asistent numit OptionGroup Builder, care poate fi lansat din meniul contextual asociat obiectului, opțiunea **Builder**. Fereastra de dialog conține:

1. *tab-ul Buttons*, prin care se pot specifica numărul de butoane și textul asociat fiecăruia;
2. *tab-ul Layout*, care permite specificarea modului de aranjare pe formular;
3. *tab-ul Value*, prin care este aleasă tabela și câmpul în care va fi memorată selecția.

Exemplu

Pe formularul de introducere a elevilor citim profilul la care s-a înscris concurentul cu ajutorul butoanelor radio afișate. Reamintim structura fișierului Elevi (nume C(20), profil N(1), clasa C(3), alte câmpuri). Pași:

1. Selectăm butonul OptionGrop de pe bara de instrumente Form Controls și îl plasăm pe formular prin tragere și plasare;
2. Deschidem fereastra Properties și specificăm la nivelul grupului proprietățile:
`Buttoncount=3,`
`ControlSource=elevi.profil;`
3. Specificăm proprietățile fiecărui buton:
`Option1.Caption='informatica',`
`Option2.Caption='sport'`
`Option3.Caption='engleza'`
4. Rulăm prin **Run**.

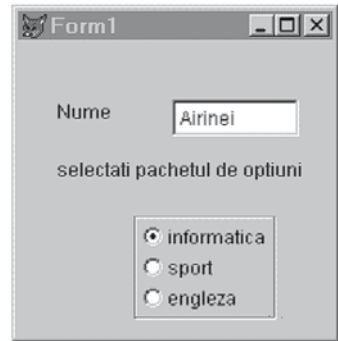


Figura 17-17: Butoane radio pe formular

Observație: Dacă alegem pentru câmpul care va primi valoarea citită prin butonul radio tipul numeric, va fi memorată poziția butonului radio selectat în cadrul grupului. Dacă tipul câmpului este caracter, va fi reținut textul explicativ asociat butonului.

g) Contoare. Obiecte Spinner

Contoarele sunt câmpuri de editare a valorilor numerice, cu verificarea apartenenței la un interval. Creșterea sau scăderea valorii se face prin clic pe butoanele corespunzătoare; trebuie fixată valoarea cea mai mare pe care o poate lua contorul (proprietatea **SpinnerHighValue**) și cea mai mică (proprietatea **SpinnerLowValue**). De asemenea, operatorul poate introduce de la tastatură valori în casetă. Vom fixa proprietățile **KeyboardHighValue** și **KeyboardLowValue**. O altă proprietate este pasul de variație, stabilit prin **Increment**. **ControlSource** specifică numele variabilei sau al câmpului unde este memorată valoarea obiectului după pierderea țintei.

Exemplu

Pentru același fișier ELEVI și formularul de citire am introdus taxa de admitere printr-un contor spinner.

Pași:

1. Deschidem formualrul precedent.

2. Plasăm obiectul și fixăm:

`SpinnerLowValue=200,000`

`SpinnerHighValue=1,500,000,Increment=1000,`

`ControlSource='elevi.taxa'`



Figura 17-18: Contor spinner adăugat la formular

h) Liste. Obiecte `ListBox` și `ComboBox`

Listele deschise sunt obiecte care se proiectează prin butonul `ListBox` din bara de instrumente Form Designer. Se prezintă ca o fereastră care afișează mai multe elemente, dintre care utilizatorul poate alege una. Listele închise se proiectează prin butonul `ComboBox` și se prezintă ca o fereastră cu un singur element și cu o bară de derulare. La deschidere este afișată toată lista și este permisă selectarea unei valori.

- Elementul selectat este memorat în câmpul pe care îl asociem proprietății **ControlSource**. Un câmp numeric va primi poziția elementului selectat, iar un câmp de tip caracter va primi chiar valoarea elementului.
- Elementele afișate pot proveni din surse diferite, specificate prin proprietățile **Rowsource** și **Rowsourcecetype**.

9a,9b,...	Elementele listei	=0 none	Sursă nespecificată; elementele listei se vor specifica dinamic la rularea formularului
elevi.dbf	Tabela de date	=1 Value	Elementele sunt enumerate
SELECT..		=2 Alias	Se preiau elementele dintr-o tabelă
A	Nume masiv	=3 SQL	Elemente date de comanda SQL
Elevi.numa	Nume de câmp	=4 Query	Elemente date de un Query
C:\alfai*.dbf?	Șablon pentru fișiere	=5 Array	Elementele sunt preluate dintr-un tablou
elevi.dbf	Numele unei tabele	=6 Fields	Elementele sunt preluate din valorile câmpului
		=7 Files	Elementele sunt nume de fișiere care verifică șablonul
		=8 Structure	Lista este alcătuită din câmpurile (structura) tablei

Alte proprietăți:

Sorted=.t. dă posibilitatea sortării alfabetice a elementelor listei la afișare.

IncrementalSearch=.T. este o facilitate deosebită pentru căutarea asistată în listele mari. Astfel, imediat ce utilizatorul introduce primul caracter, cursorul se plasează pe linia elementului care începe cu acel caracter.

■ De obicei este selectat un singur element dintr-o listă. Dacă trebuie selectate mai multe, folosim proprietatea **MultiSelect=.t.** Selectarea se face cu mouse-ul (ținând tasta Ctrl apăsată și executând clic pe elementele respective) sau cu tastatura (Ctrl+Space). Pentru a testa dacă o opțiune a fost selectată se folosește proprietatea **Selected**, un vector boolean care are pentru fiecare element al listei valoarea .T. dacă a fost selectat și .F. dacă nu. Numărul elementelor listei este reținut în proprietatea **ListCount**, iar valorile propriu-zise sunt reținute în proprietatea **List**.

Exemple

Exemplul 1: Liste deschise cu diferite surse

Folosim o listă deschisă pentru alegerea clasei noului elev. Elevi (nume, adresa, clasa, media...).

Pași:

1. Executăm clic pe butonul **ListBox** din bara de instrumente Form Controls și proiectăm poziția și dimensiunea pe suprafața de lucru.
2. Definim proprietățile. Pentru câmpul unde se face citirea, definim **ControlSource='elevi.clasa'**. Pentru specificarea elementelor listei, alegem din trei variante pe care le vom explica pe rând.

Varianta 1. Elementele listei sunt trecute manual la proiectarea obiectului. Folosim proprietățile: **RowSourceType=1; RowsSource= '12a,12b,12c,12d'**.

Varianta 2. Elementele listei sunt preluate dintr-un tablou numit A.

Proprietățile sunt **RowSourceType=5 Array; RowsSource=A**; Adăugăm la metoda **Load** a formularului comanda **DECLARE A[4]**, iar în metoda **Init** vom inițializa elementele tabloului: **a[1]='12a', a[2]='12b'** etc.

Varianta 3. Elementele listei sunt preluate dintr-un fișier.

Extragerea elementelor listei se face din câmpul **clasa** al tabelii **Clase.dbf** (clasa C(3), diriginte C(10), sala c(4)). Pentru aceasta deschidem fișierul în **Data Environment** (deschidem fereastra prin clic pe butonul din **Form Designer**, apoi din meniul contextual alegem **ADD**). Proprietățile listei sunt:

RowSource='clase.clasa', RowSourceType=6.

Varianta 4. Lista este formată din mai multe câmpuri unite într-o expresie.

Afișăm numele dirigintelui și sala împreună cu numele clasei, dar extragem doar clasa. Proprietăți: **RowSourceType=2 alias RowsSource= 'clase.clasa+ clase.diriginte+ clase.sala'**

3. După fixarea proprietăților, lansăm formularul prin **Run** și, bineînțeles, deschidem fereastra **Browse** să verificăm corectitudinea citirilor.

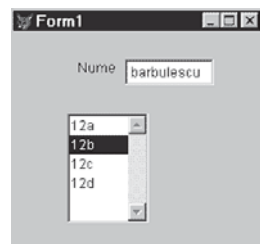


Figura 17-19:
Listă deschisă
introdusă în formular

Exemplu 2: Listă închisă. Elementele listei sunt nume de fișiere

Să presupunem că avem Fisiere.dbf, o tabelă cu toate fișierele unui grup de proiectare cu structura (nume-fisier C(20), autor C(30)).

Vrem să proiectăm un formular pentru adăugarea în tabela Fisiere a numelui unui fișier (numai de tip dbf) ales dintr-o listă închisă. Observați formularul din figură.

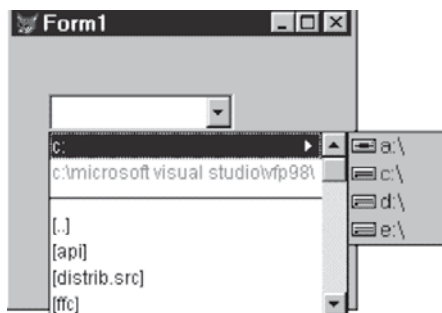


Figura 17-20: Listă închisă în formular

Pași:

1. Deschidem în fereastra Data Environment tabela Fisiere.
2. Scriem comanda Append blank pentru metoda Init a formularului, care permite adăugarea unui nou articol în Fisiere.
3. Proiectăm obiectul prin butonul ComboBox ales de pe bara de instrumente și poziționarea sa pe formular.
4. Stabilim proprietățile esențiale: `ControlSource='fisiere.nume_fis'`, `RowSource= *.dbf`; `RowSourceType=6 Files`.
5. Rulăm prin Run și urmărim noul articol în fereastra Browse.

Exemplu 3: Liste deschise cu selecții multiple

Ne propunem să selectăm dintr-o listă deschisă care afișează numele tuturor fișierelor cu extensia .doc din directorul C:\vfp6 mai multe elemente și să le adăugăm în tabela Fisiere.dbf în câmpul Nume_fis.

Pași:

1. Plasăm obiectul ListBox pe formular;
2. Fixăm proprietățile: `ControlSource='fisiere.nume_fis'`
`RowSource='C:\vfp6*.doc'`; `RowSourceType=6 Files`;
`MultiSelect=.T.`;
3. Activăm fereastra de proprietăți a formularului și scriem pentru obiect metoda Lostfocus (arătam ce vom face după ce se termină lucrul cu formularul):

Procedura Lostfocus

```
select fisiere
for i=1 to thisform.list1.listcount
if thisform.list1.selected[i]
append blank
replace nume;
with thisform.list1.list[i]
endif
endfor
endproc
```

Poziționare pe tabela destinație

Listcount – numărul de elemente din listă

Selected – vectorul asociat listei cu valoare .t.=selectat

List – vectorul cu valorile listei

List1 – este numele obiectului

Editarea rapidă: List Box Builder

Pentru editarea rapidă a obiectelor de tip listă se folosește asistentul List Box Builder, lansat din meniul contextual al obiectului, opțiunea Builder.

1. Tab-ul List Items permite precizarea sursei elementelor, fie introduse manual, fie dintr-un tablou, fie dintre câmpurile unei tabele.

2. Tab-ul Style permite precizarea aspectului obiectului (Special Effect), a numărului de linii (corespunde proprietății Height), dacă dorim căutarea asistată (corespunde proprietății IncrementalSearch).

3. Tab-ul Layout oferă o previzualizare a listei. Putem să modificăm direct lățimea coloanei (coloanelor) sau să dispunem ajustarea lor automată (adjust).

4. Tab-ul Value ne ajută să fixăm coloana pentru care dorim să i se memoreze valoarea selectată (proprietatea BoundColumn). Proprietatea ControlSource este setată prin secțiunea FieldName.

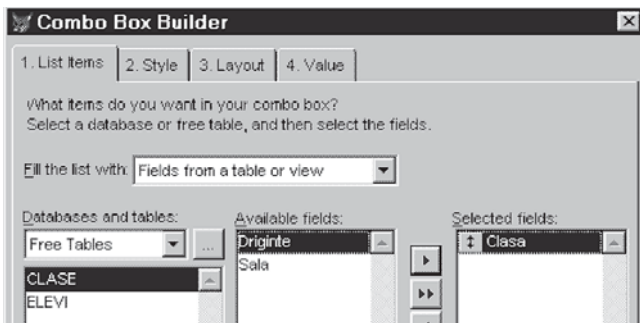


Figura 17-21: Fereastra Combo Box Builder

Exemple

Un exemplu de formular pentru actualizarea datelor

Să presupunem că pentru o bază de date creată la banca „Ion Popescu” informațiile despre comisioanele practicate pentru conturi curente, depozite, alte operații bancare se rețin în tabela Comisioane(tip_comision, denumire, procentul aplicat drept comision).

Realizăm un formular care permite adăugarea unui nou articol, editarea procentului sau ștergerea articolului curent. Articolul curent este determinat prin selectarea unui element din listă. Lista afișează liniile tabelii Comision. Odată selectată o linie, sunt afișate în zone separate tipul, denumirea și procentul și este permisă editarea lor. La închiderea formularului dorim afișarea unui mesaj.

Observați proiectul formularului și proprietățile/metodele asociate formularului și obiectelor de control.

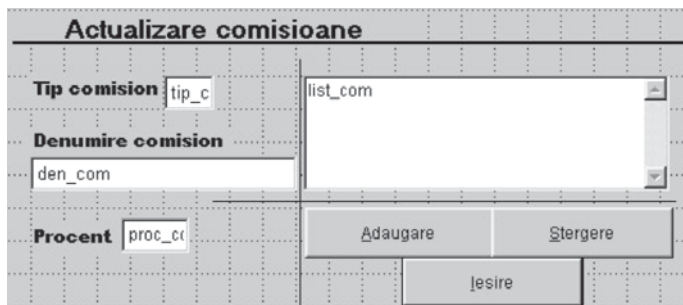


Figura 17-22: Formularul de actualizare a comisioanelor

<pre> Form1.destroy thisform.refresh if mess("Vrei sa inchizi?",36, "Inchidere") = 6 thisform.release else if mess("Vrei sa inchizi?",36,"Inchidere")=7 endif endif Form1.caption='actualizare' </pre>	<pre> Form1. init if !used('comision') use comision in 0 endif set deleted on </pre> <hr/> <pre> Form1. release set deleted off sele comision pack </pre>
--	---

<pre> Tip_comision.ControlSource='comision.tipcomision' lesire. caption='iesire' lesire. click thisform.release sterger.click sele comision dele skip-1 thisform.refresh </pre>	<pre> Dencomision. lostfocus if thisform.comml.tag='t' reindex thisform.list_com.refresh thisform.comml.tag='f' else thisform.list_com.refresh endif </pre>
---	--

<pre> Procent_comision.click if thisform.comml.tag='t' reindex thisform.list_com.refresh thisform.comml.tag='f' else thisform.list_com.refresh endif </pre>	<pre> List_comision.click sele comision go top locate for; tipcomision=val(thisform.list_com.value) thisform.tip_com.refresh thisform.den_com.refresh thisform.proc_com.refresh </pre>
---	--

<pre> list_comision Rowsource="comision.tipcomision,denco mision,; procent" RowSourceType="fields" Tabindex=4 Columncount=3 Name=list_com </pre>	<pre> Aaugare.click go bottom append blank thisform.comml.tag='t' thisform.refresh </pre>
--	---

i) Grile. Obiecte Grid

Grilele sunt obiecte de interfață de tip container, proiectate prin butonul Grid de pe bara de instrumente Form Controls, apoi trasarea zonei ocupate de obiect pe suprafața formularului. Grilele sunt folosite pentru introducerea mai comodă a articolelor într-o tabelă/o vedere.

Grilele au în componență mai multe obiecte de intefată plasate într-un tabel, asemeni tabelului Browse. La intersecția *liniilor* cu *coloanele* se află *celula* folosită pentru citirea/vizualizarea valorilor preluate dintr-o variabilă sau un câmp al unei

tabele/vederi. *Antetul grilei* este folosit pentru denumirile coloanelor. Există o coloană pentru ștergere și o coloană pentru selectarea liniilor. Grila este plasată într-o fereastră cu bare de derulare verticale și orizontale atunci când dimensiunea matricii depășește dimensiunea zonei proiectate.

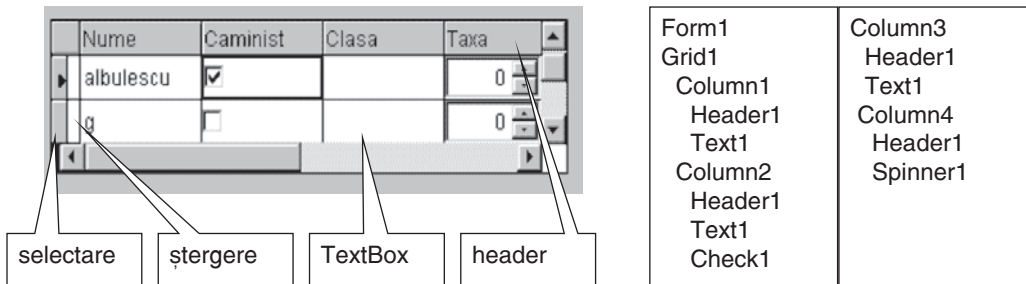


Figura 17-23: Elementele unei grile

Corespunzător acestor componente, obiectul Grid este compus într-o structură arborescentă din mai multe obiecte Column, care, la rândul lor, sunt compuse din **Header** (anteturile pentru titlul coloanei) și, implicit, **Textbox**-uri (casete de text pentru afișarea/editarea informației de pe coloană). Dacă dorim, celula poate avea alt obiect pentru editarea valorii (de exemplu, **CheckBox** ca în coloana Caminist, **Spinner** ca în coloana Taxa).

Dacă nu specificăm proprietatea **Name** pentru nici un obiect din exemplu, sunt asociate în mod implicit nume de obiecte.

Fiecare dintre aceste componente are proprietăți și metode.

■ **Referirea la o componentă a grilei** se face folosind calificarea.

De exemplu, dacă dorim ca zona de editare a coloanei Clasa să aibă textul centrat, vom scrie:

```
Form1.Grid1.Column3.Text1.Alignment=2
```

Întreaga grilă poate fi referită prin metoda **SetAll** care determină atribuirea unei proprietăți pentru întregul grup. Exemplu:

```
Form1.Grid1.SetAll("fontName", "arial")
```

De asemenea, se poate face referire la o întreagă coloană știind că grilei i se asociază un vector special **Columns**. Exemplu:

```
Form1.Grid1.Columns[1].Datasource='elevi.nume'
```

■ **Numărul de coloane ale unei grile** este dat de proprietatea **Columncount**. Dacă există o tabelă deschisă, toate coloanele ei vor popula grila și implicit **columncount=-1**.

Pentru a indica sursa de date a întregii grile se folosește proprietatea:

RowSourceType – care poate avea valorile:

- =0-Table sau =1-Alias – când datele sunt preluate dintr-o tabelă, iar **Rowsource** va trebui să conțină numele tabelii;
- =2-Prompt – când datele sunt furnizate la execuție de către utilizator;
- =3-Query – când datele sunt preluate dintr-un fișier Qpr;
- =4-SQL – când sursa de date este rezultatul unei comenzi SQL, iar **Rowsource** va cuprinde comanda **Select**.

- Pentru a indica **sursa de date a unei coloane** se folosește proprietatea **ControlSource** cu numele câmpului precedat de alias.

De obicei, datele sunt preluate dintr-o tabelă și este deci util ca la proiectarea grilei să avem deschisă tabela, astfel încât la completarea proprietății **RowSource** sau **ControlSource** să putem alege dintr-o listă valoarea dorită.

■ Operații cu datele unei grile (proprietăți la nivelul grilei):

1. Adăugarea datelor – se permite prin **AllowAddNew=.T.** astfel încât la ultima linie, dacă se apasă săgeata în jos, se mai adaugă o linie.
2. Ștergerea liniilor – se face prin marcarea lor în coloana de ștergere, asemeni ferestrei Browse. Dacă dorim dezactivarea acestei coloane, deci și a posibilității de ștergere, fixăm **DeleteMark=.F.** Setarea implicită este **.t.**
3. Modificarea câmpurilor este implicită, dar dacă dorim să permitem doar vizualizarea, stabilim **ReadOnly=.T.** Proprietatea se poate plasa și la nivelul unei coloane, dacă dorim numai protecția acesteia.

Editarea grilei este permisă prin meniul contextual asociat grilei, apoi selectând opțiunea **Edit**. Se observă schimbarea conturului grilei la acceptarea editării. Prin poziționarea pe o coloană se pot schimba proprietățile acesteia în fereastra Properties. Se pot modifica dimensiunile coloanei prin tragerea cu mouse-ul a benzilor separatoare. Pentru ștergerea unei coloane se apasă tasta **Delete**. Adăugarea unei coloane se face prin schimbarea la nivelul grilei a proprietății **ColumnCount** la noul număr și completarea manuală a header-ului și a sursei de date pentru TextBox.

■ Editarea rapidă a unui obiect de tip grilă prin Grid Builder

O metodă de construire rapidă a unei grile este oferită de asistentul Grid Builder, activat prin meniul contextual al grilei, opțiunea Builder.

1. Tab-ul **Grid Items** permite deschiderea/selectarea bazei de date, a tabelului sursă și precizarea câmpurilor care vor forma coloanele grilei.
2. Tab-ul **Style** setează aspectul grilei.
3. Tab-ul **Layout** permite fixarea obiectelor asociate. Pentru fiecare coloană se stabilește titlul (Caption) și tipul obiectului (Control type). Se poate alege dintre TextBox, CheckBox, Spinner etc. De asemenea, se pot ajusta dimensiunile celulelor (proprietățile **Height**, **Width**).
4. Tab-ul **Relationship** conține întrebări pentru a permite crearea unei grile cu două tabele legate.

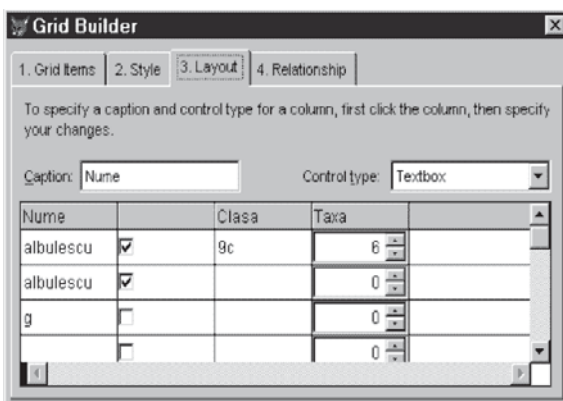


Figura 17-24: Fereastra Grid Builder

■ Proiectarea rapidă a obiectului grilă prin Data Environment

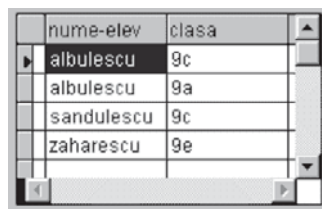
O altă metodă de a crea rapid grile este folosirea obiectului Data Environment, care conține toate tabelele și vederile deschise pe parcursul proiectării unui formular. Poate fi lansat din bara de instrumente Form Designer sau din meniul View. Ne poziționăm pe titlul tabelului căreia dorim să-i proiectăm o grilă și executăm tragere și plasare pe suprafața de lucru a formularului. Desenăm o grilă pentru toate câmpurile tabelului, având ca antet chiar numele câmpului și ca obiecte de editare casete de text.

Exemplu

Realizăm o grilă de actualizare a numelui și clasei elevilor din tabela Elevi (nume C(10), adresa M, clasa c(3), media N(5,2), taxa N(5), caminist I, dn D, foto G). Pași:

Varianta 1: Proiectare pas cu pas.

1. Deschidem tabela Elevi în fereastra Data Environment.
2. Proiectăm grila prin clic pe butonul Grid de pe bara de instrumente Form Controls; poziționăm și dimensionăm obiectul direct pe suprafața de lucru.
3. Intrăm în editarea containerului prin meniul contextual al grilei (opțiunea Edit).
4. Fixăm proprietățile fiecărui obiect:



nume-elev	clasa
albulescu	9c
albulescu	9a
sandulescu	9c
zaharescu	9e

Figura 17-25: Grila de actualizare a numelui și clasei

Pentru Grid1:
ColumnCount=2,
AllowAddNew=.t.

RowSource='elevi'

RowSourceType=0

La Metoda Lostfocus

Browse

Pentru Column1:
ControlSource='elevi.n
ume'

Pentru Header1:
Caption='nume-elev'

Pentru Column2:
ControlSource='elevi.c
lasa'

Pentru Header1:
Caption='clasa'

Varianta 2: Proiectarea rapidă prin Data Environment.

1. Deschidem tabela în Data environment și ne poziționăm pe bara de titlu a tabelului.
2. Executăm tragere și plasare până la suprafața formularului.
3. Alegem din meniul contextual (clic dreapta) opțiunea **Edit**.
4. Ne poziționăm cu mouse-ul pe coloana a doua (coloana corespunzătoare adresei, de care nu avem nevoie) și o ștergem apăsând tasta **Delete**.
5. În fereastra Properties fixăm **ColumnCount=2** (cu această ocazie ștergem și coloana Media) și **AllowAddNew=.t.** (pentru că valoarea implicită este **.F.**).
6. Executăm de probă prin clic pe **Run**.

Varianta 3. Proiectarea rapidă prin Grid Builder.

Ne propunem realizarea unei grile pentru actualizarea tabelului Elevi, dar care să aibă pentru câmpul Caminist un Checkbox, pentru câmpul Taxă un Spinner, iar pentru câmpul Adresa un obiect de tip Editbox.

Pași:

1. Ne poziționăm în fereastra de lucru a generatorului de formulare și executăm tragere și plasare de la butonul Grid de pe bara de instrumente Form Controls pe formular.
2. Deschidem meniul contextual prin clic dreapta și alegem **Builder**.
3. Răspundem la întrebări și rulăm prin **Run**.

Observație:Proiectarea unei grile prin Data Environment asociază coloanelor doar TextBox-uri.

O metodă de a pune alte obiecte drept coloane ale grilei este adăugarea obiectelor prin codul metodei Init asociate grilei. De exemplu, pentru o grilă unde dorim pe coloana corespunzătoare câmpului caminist un obiect de tip **CheckBox**, iar pe coloana unde este taxa un **spinner** scriem codul următor:

```
**Procedure Init ***pentru grid
This.column2.addobject('check1','checkbox')
This.column4.addobject('spinner1','spinner')
This.column2.currentControl='check1'
This.column4.currentControl='spinner'
This.column2.check1.visible=.T.
This.column2.check1.caption='elevi.caminist'
This.column4.spinner1.spinnerhighvalue=2000
This.column4.spinner1.spinnerlowvalue=100
This.column4.spinner1.increment=100
This.column4.spinner1.controlSource='elevi.taxa'
```

O altă metodă de a schimba obiectul de editare² este următoarea: deschidem fereastra Browse asociată formularului și căutăm linia corespunzătoare coloanei căreia dorim să-i schimbăm obiectul de editare. Schimbăm valorile din câmpurile Class, BaseClass și ObjName pentru a referi noul obiect asociat coloanei și ștergem câmpul **Properties**.

Închidem fișierul obiectului formular prin **Close all**. Revenim în constructorul de formulare prin **modify form** și edităm formularul.

j) Seturi de Pagini. Obiecte Page Frame

Paginile și seturile de pagini sunt elemente de interfață care asigură centralizarea informațiilor în același formular și sistematizarea lor pe pagini distincte. În literatura de specialitate am întâlnit denumirea de tab-uri. Se mai numesc și pagini alternative, pentru că la un moment dat o singură pagină este activă. Definirea unui set de pagini se face prin clic pe butonul **Page Frame** de pe bara de instrumente Form Controls și trasarea zonei ocupate pe formular. Fiecare pagină este un obiect numit **Page**.

² Propus de Gabriel și Mihai Dima în Bazele Visual FoxPro 5.0

La nivelul setului de formulare se definesc proprietățile:

- Numărul de pagini – proprietatea **PageCount**;
- **Dimensiunea paginilor – PageHeight (înălțimea) și PageWidth (lățimea)**;
- Modul de afișare a titlurilor de pagină **Tabstretch** – valoarea 0 (multiple rows), dacă sunt mai multe pagini și anteturile lor nu încap pe un singur rând, se creează două linii cu anteturi; valoarea 1 (single row); se ajustează titlurile paginilor pentru a încăpea pe o singură linie;
- **Caption** – textul explicativ la nivelul paginii (antetul);
- **Activepage** – fixează pagina activă (la un moment dat o singură pagină este activă);
- **PageOrder** – fixează numărul de ordine al paginii (în mod implicit, sunt numerotate în ordinea proiectării).

O metodă importantă la nivelul paginii este **Activate**, care va fi executată la activarea paginii. Putem introduce aici secvențe care reîmprospătează datele prin citirea lor de pe disc în cazul în care acestea s-au modificat.

Obiectul PageFrame este un container care cuprinde mai multe pagini, la rândul lor containere, organizate după un model arborescent. Referirea la un obiect se face prin indicarea întregii ramuri părinte.

Utilitarul acordă nume implicite obiectelor, după cum bine știm, prin numerotarea obiectelor din aceeași clasă. Deci, dacă avem trei pagini, ele vor primi numele Page1, Page2, Page3.

O altă modalitate de referire a paginilor este proprietatea **Pages** – un vector care are ca elemente paginile componente ale setului. Referirea la un obiect în pagină se poate face și prin numele obiectului sau prin proprietatea **Controls** – un vector care are ca elemente obiectele de control definite în pagina respectivă.

Exemplu

Acordarea numelui paginii 3

```
Pageframe1 . Page3 . caption = ' Listare '  
sau  
Pageframe1 . Pages [3] . caption = ' Listare '
```

Activarea paginii 1

```
Pageframe1 . page [1] . ActivePage = . t .
```

Pe prima pagină avem un TextBox5 (presupunem că este al 12-lea obiect pe pagină) căruia îi asociem metoda SetFocus.

```
Pageframe1 . page1 . textbox5 . setfocus  
sau  
Pageframe1 . page [1] . controls [12] . setfocus
```

Odată configurat, obiectul Set de pagini poate fi editat separat, accesând meniul contextual, opțiunea Edit sau selectându-l în arborele de obiecte a paginii dorite.

Vom exemplifica modul de proiectare a unui formular cu seturi de pagini pornind de la obiectele de tip grilă realizate anterior, asupra cărora nu ne vom opri. Deci pe prima pagină este o grilă pentru actualizarea elevilor, iar pe cea de-a doua o grilă pentru actualizarea claselor. Trecem de la o pagină la alta, iar când dorim încheierea lucrului apăsăm butonul de comandă: „gata“.

Pași:

1. Definim colecția de pagini pornind de la butonul Page Frame, trasând zona ocupată de acesta pe formular.
2. Definim butonul CommandButton pe formular în afara setului.
3. Intrăm în editare (din meniul contextual, opțiunea Edit) și fixăm proprietățile fiecărei pagini. Astfel:

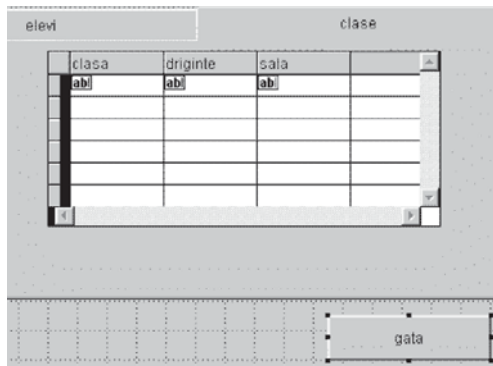


Figura 17-26: Proiectarea formularului cu seturi de pagini

Pagecount=2	Caption='elevi'	Caption='clase'	Caption='gata'
			Procedure click
			Thisform.release

4. Plasăm pe pagina 1 un obiect grilă pentru Elevi și pe pagina 2 un obiect grilă pentru tabela Clase, pentru care nu mai arătăm proprietățile.
5. Rulăm prin comanda Run.

k) Imagini. Obiecte Image

Formularele de culegere a datelor sau de vizualizare sunt mai atractive când conțin desene, imagini, sigle. Proiectarea într-o zonă a formularului a obiectelor imagine, care sunt de fapt fișiere .bmp, se face prin clic pe butonul Image de pe bara de instrumente Form Controls prin tragere și plasare, apoi conturarea zonei ocupate de imagine. Proprietăți:

- **Picture** – specifică numele fișierului sursă;
- **BorderStyle=1**: imaginea este încadrată în chenar și ...=2: contrar;
- **Stretch** – fixează modul de afișare a imaginii și poate lua valorile:
 - =0 Clip – se păstrează dimensiunile inițiale ale imaginii sau se decupează imaginea;
 - =1 Isometric – imaginea se încadrează în zonă păstrând proporțiile;
 - =2 Stretch – imaginea este ajustată prin deformare la zonă.
- Dintre evenimente, se pot enumera Click, Drag (tragere cu mouse-ul) etc.

l) Ceas. Obiecte Timer

Ceasul ca obiect de interfață permite măsurarea intervalelor de timp la care trebuie executate anumite acțiuni, de exemplu, afișarea unor mesaje care solicită alegerea unei opțiuni. Dacă utilizatorul nu alege în timpul disponibil, va fi folosită o variantă implicită.

Definirea obiectului începe prin aducerea butonului timer de pe bara de instrumente Form Controls pe suprafața de lucru. Proprietăți:

- **Interval** – fixează intervalul de timp între două evenimente Timer. Intervalul este specificat în miimi de secunde.

Metoda Timer specifică acțiunea care va fi executată la expirarea timpului.

Exemplu

Exemplul pe care îl folosim „plimbă“ o imagine de la dreapta formularului la stânga și invers. Inițial se afișează doar poza din stânga. La clic pe buton apare în dreapta timp de 2 secunde, apoi se mută din nou în stânga. Pași:

1. Proiectăm pe formular obiectul Timer, apoi obiectul Image1 (pe care apoi îl copiem și îl denumim Image2) și Command1.
2. Scriem proprietățile și metodele (conform tabelului următor):

Timer1	Image1 (stânga)	Image2 (dreapta)	Command1
Interval=2000 Procedure Timer thisform.image1.visible=.t. thisform.image2.visible=.f.	Visible=.t.	Visible=.f.	Procedure click thisform.image1.visible=.f. thisform.image2.visible=.t.

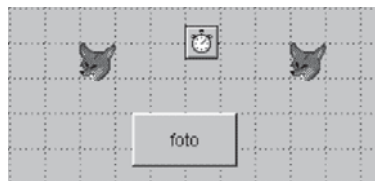


Figura 17-27: Obiect Timer în formular

m) Obiecte de tip OLE

Tehnica OLE este deja cunoscută din lecțiile anterioare de programare a aplicațiilor Windows, permițând interacțiunile dintre aplicațiile Windows prin intermediul obiectelor suportate de acestea. Obiectele pot fi legate (*linked*) sau incluse (*embedded*). La execuție pot fi editate folosind aplicația care le-a generat.

Controalele ActiveX, memorate pe disc sub forma fișierelor **.OCX**, pot fi incluse direct într-un formular, ca și obiectele Fox. Ele sunt obiecte OLE, în sensul că sunt create de alte aplicații. Important este că au propriul set de proprietăți și evenimente/metode pe care le putem modifica la proiectare. Din acest punct de vedere, FoxPro este o aplicație client, pentru că nu oferă obiecte altor aplicații.

Obiecte de tip OLE Container control

Aceste obiecte sunt proiectate prin butonul OLE Control, care deschide fereastra de dialog **Insert Object**. Poate fi creat un nou obiect sau poate fi inclus unul existent.

Obiecte de tip OLE Bound Control

Aceste obiecte editează câmpurile General din tabele de date. Sunt proiectate prin butonul OLE Bound Control și, după specificarea zonei ocupate de obiect, sunt definite proprietățile. **ControlSource** desemnează numele câmpului de tip general care dorim să fie editat prin formular.

Dintre proprietățile asociate ambelor obiecte OLE amintim:

- **Autoactive** – specifică modul de activare a obiectului. Poate lua valorile:

=0 Manual înseamnă activare prin cod folosind metoda **Doverb**.

=1 Gotfocus – obiectul este activat când este selectat cu mouse-ul sau cu tastatura.

=2 DoubleClick – obiectul este activat prin dublu clic.

=3 Automatic – obiectul este activat automat la afișarea formularului.

Metoda **doverb** execută o funcție specifică aplicațiilor server OLE (de exemplu, **EDIT, SAVE, OPEN**).

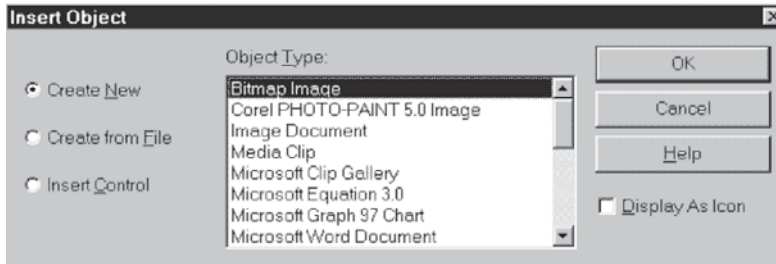


Figura 17-28: Fereastra Insert Object

De exemplu, dorim ca pe formularul de culegere a datelor despre un elev, în afară de introducerea numelui și a pozei preferate, să permitem operatorului să-și „deseneze” semnătura. Folosim fișierul Elevi(nume C(20), poza G,...).

Pași:

1. Deschidem fișierul Elevi.
2. Proiectăm formularul cu obiectele Label.
3. Pentru TextBox indicăm **ControlSource = 'elevi.nume'**.
4. Pentru semnătură folosim obiectul ActiveX OLE de pe bara de instrumente Form Designer.
5. Alegem din fereastra Insert Object tipul Bitmap Image și bifăm Create New.
6. În ecranul de proiectare deschis desenăm cele trei litere.
7. Închidem fereastra de proiectare și revenim în mediul formularului.
8. Pentru poză, alegem butonul OLE Bound Control de pe bara de instrumente Form Controls și indicăm legătura cu sursa de date, deci proprietatea **ControlSource='elevi.poza'**.



Figura 17-29: Obiect OLE introdus în formular

o) Linii, chenare, cercuri. Obiecte Line și Shape

Obiectele grafice, ca liniile și chenarele, îmbunătățesc aspectul unui formular.

Proprietăți ale liniilor:

- **BorderColor** – culoarea;
- **BorderStyle** – modul de desenare (0=transparent, 1=continuu, 2=linie întreruptă etc.);
- **BorderWidth** – grosimea în pixeli;
- **LineSlant** – orientarea.

Forma prestabilită a obiectelor formă (*shape*) este dreptunghiul. Proprietățile sunt:

- **SpecialEffect** =1 dă un aspect tridimensional;
- **FillStyle** – stilul obiectului ...=1 transparent, ...=2 opac;
- **FillColor** – culoarea de umplere;
- **BorderColor** – culoarea marginii;
- **Curvature** – gradul de rotunjire (între zero și 99 – pentru cerc).

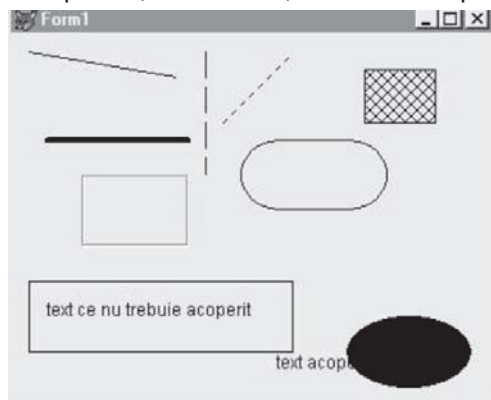


Figura 17-30: Linii, chenare și cercuri introduse în formular

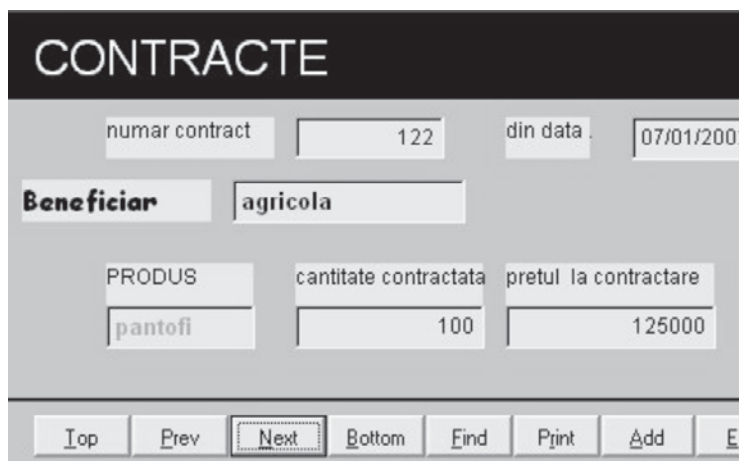


Figura 17-31: Obiecte formă adăugate în formular

Proiectarea interfeței folosind Form Wizard

FoxPro conține un vrăjitor (aplicație wizard) pentru realizarea rapidă a formularelor cu aspect profesionist. Observați un astfel de formular care permite nu numai căutarea și poziționarea în tabela de date, ci și actualizarea sau afișarea la imprimantă.

Lansăm FORM WIZARD selectând **Tools, Wizard, Form**. Form Wizard permite crearea rapidă a unui program direct executabil (**fis.scx**) pentru editarea câmpurilor dintr-o tabelă sau din două tabele legate între ele. Vom prezenta pe rând cele două posibilități.



Fereastra 1 deschide baza de date, tabela dorită și selectează câmpurile care vor fi editate prin formular.



Fereastra 2 permite specificarea unui model de formular din cele prestabilite. Printre opțiuni sunt: Standard – afișarea verticală a câmpurilor; Chiseled – rândurile au un efect de subliniere 3D etc. Pot fi plasate butoane de derulare cu text sau imagine.



Fereastra 3 permite alegerea, dacă este cazul, a câmpurilor după care se va face ordonarea automată a înregistrărilor pentru vizualizare.



Fereastra 4 permite salvarea și rularea imediată sau ulterioară a formularului sau modificarea acestuia cu Form Designer.

O altă posibilitate oferită de asistent este **editarea simultană a datelor din două tabele legate între ele**. Câmpurile din prima tabelă formează antetul formularului, iar cele din a doua tabelă sunt dispuse pe coloane, asemeni ecranului Browse într-un subformular. Se poate trece de la un articol al primei tabele la altul. Automat, în subformularul asociat tabelii copil se pot edita doar datele legate de articolul curent din tabela părinte.

În concluzie

Formularul este un element central al unei interfețe Windows care poate fi proiectat atât prin comenzi Visual FoxPro (obiectul aparține clasei container Form) cât și interactiv, prin utilizarea instrumentului de proiectare Form Designer și a asistenților (Builder) asociați.

Formularele pot fi folosite ca ferestre de prezentare a aplicației sau ca panou de bord. Principalul lor rol este însă vizualizarea și editarea datelor din unul sau mai multe tabele.

Trebuie avute în vedere câteva **principii**³ la proiectarea interfețelor utilizator, astfel încât orice utilizator, chiar dacă nu a folosit niciodată aplicația, să fie în măsură să înțeleagă și să lucreze cu programul aplicației.

³ Bob Grommes, FoxPro 2.5 pag. 395

Aceste principii sunt:

- Prezentarea tuturor informațiilor necesare utilizatorului trebuie făcută în mod intuitiv. Astfel, afișarea stinsă sau dezactivată în fereastră a unor obiecte care nu sunt disponibile în momentul respectiv oferă, un plus de siguranță utilizatorului. Este indicat ca textele să fie clar afișate, folosind diverse culori.
- Gruparea logică a meniurilor și obiectelor de interfață.
- Autodocumentarea fiecărei acțiuni, opțiuni etc.
- Folosirea standardelor pentru numele butoanelor, de exemplu; Cancel are aceeași semnificație peste tot, nu este necesar să schimbați numele sau să îl înlocuiți cu Abandon).
- Informarea permanentă a utilizatorilor privind ceea ce se întâmplă. Ori de câte ori se apasă un buton sau o tastă, trebuie să se întâmple ceva, de aceea pentru operațiile lungi afișați un indicator de proces (bare de progres, care prezintă procentajul realizat din proces) sau mesaje de informare.
- Avertizarea sonoră în cazurile de eroare este indicată!
- Mesajul de operație ilegală trebuie particularizat precizând exact unde și ce s-a greșit.
- Deorece în maniera procesării conduse de evenimente a aplicației sunt deschise simultan pe ecran mai multe ferestre, este important ca fiecare mesaj sau fereastră de dialog să aibă referire la fereastra sau procesul care a cauzat intervenția.
- Este bine ca utilizatorului să i se permită personalizarea mediului de lucru: culori, viteza mouse-ului, starea difuzorului și orice altceva este posibil. Acest lucru dă aplicației un aspect profesional dar, în același timp, prietenos față de utilizatori.
- Pentru evitarea cazurilor de ștergeri accidentale sau de deteriorare a informațiilor este recomandată plasarea operației într-un loc mai greu accesibil, fără tastă directă etc. De asemenea, se va cere confirmarea utilizatorului la orice operație distructivă. Dacă totuși s-a executat o manevră greșită trebuie avută în vedere posibilitatea refacerii datelor (prin comanda UNDO).



sarcini de laborator

I. Sarcini pentru proiectarea unui formular cu parametri

1. Construiți un formular care să primească din exterior (ca parametru de intrare) un număr între 1 și 7, să-l afișeze pe formular într-un chenar dacă este par și fără chenar dacă este impar și să întoarcă numele zilei corespunzător acestui număr, tot ca parametru în variabila Nume. Afișați apoi această variabilă!
2. Imaginați o variantă de rezolvare pentru ca formularul să primească două numere: primul, dacă este zero, va determina considerarea celui de-al doilea ca număr de zi (din săptămână), iar dacă primul număr este 1, cel de-al doilea va reprezenta numărul unei luni (din an). Pe formular vor fi afișate cele două numere primite din exterior cu culori diferite și alăturat numele zilei sau a lunii. Formularul va returna un mesaj de tipul „ziua este...” sau „luna este...” completând

numele obținut numai dacă este satisfăcută condiția de validare. Atenție, nu se citesc cele două numere: se primesc drept parametru!

3. Modificați formularul pentru introducerea datelor în ELEV1 astfel încât să afișeze și adresa (presupunem câmpul adresa pe 30 caractere, din care primele sunt pentru localitatea elevului) și dacă elevul nu este din Iași (sau o localitate pe care s-o dați ca parametru de intrare), atunci să fie automat considerat căminist și operatorul să aibă acces la completarea valorii Taxa.
4. Încercați să închideți formularul atunci când s-au terminat articolele. Vă aduceți aminte de metoda Release a formularului? Unde va fi plasată?

II. Sarcini pentru proiectarea pas cu pas a obiectelor de control

Odată cu recepția produselor de la furnizori se înregistrează informațiile de pe factura însoțitoare în fișierul PRODUSE(furniz C(20), produs C(20), tip_mf C(5), pi N(7), tva N(7), tva N(7), adaos N(7), impozit N(7)), unde: furniz=numele furnizorului; tip_mf=tipul mărfii (băuturi, țigări, cafea, alte alimente), pentru care se percepe impozit diferit; pi=preț de intrare al produsului de pe factură; tva=suma calculată conform procentului de tva aplicat asupra prețului de intrare; adaos=suma adaos calculată prin aplicarea procentului de adaos fixat pentru produs de societate; impozit=suma calculată ca impozit pentru cafea, țigări, băuturi etc., conform cotelor de impozitare legale.

Proiectați formularul de introducere a datelor de pe factură folosind tipuri diferite de obiecte de control!

III. Sarcini pentru construirea unei interfețe a aplicației BIBLIOTECA

Pentru baza de date Biblioteca au fost construite tabelele Carti (isbn, titlu, cod_autor, editura, colectia), Autori (cod_autor, nume, data_n, data_d, naționalitate), Cititori (cod_cit, nume, adresa, ocupația, telefon).

1. Realizați un grup de **declanșatori** care permite afișarea tablei Carti pe ecran, la imprimată sau într-un fișier. Analizați procedura Click alăturată! Unde ar trebui plasată? La nivelul formularului, al grupului sau al primului buton? Credeți că mai trebuie schimbat ceva? Cum să asociem o pictogramă numai unui singur buton?
2. Construiți un formular care permite introducerea noilor cărți prin preluarea dintr-o listă deschisă a numelor de autori (elementele vor fi sortate alfabetic și vor proveni din valorile câmpului nume al tablei Autori), prin preluarea numelui editurii dintre elementele unui masiv, iar pentru colecție lista va fi completată manual.
3. Pentru baza de date BIBLIOTECA construiți un formular ca și cel din imagine, care permite editarea și adăugarea cărților.
4. Construiți un formular de vizualizare a cărților unui autor X, dat ca parametru la intrarea în formular.
5. Modificați formularul anterior pentru ca afișarea cărților să se facă pentru o anumită colecție sau editură.

```
Procedure click
Do case
  Case this.value=1
    List
  Case this.value=2
    List to print
  Otherwise
    List;
    to file x.txt
Endcase
Endproc
```

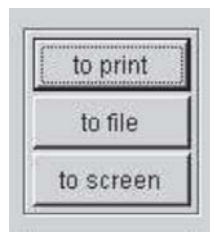


Figura 17-32:
Grupul de
declanșatori

- Proiectați un formular de culegere a datelor cu două pagini. Prima va permite introducerea unei cărți și, dacă autorul respectivei cărți nu este înregistrat în tabela autori, se va deschide pagina de introducere a autorului. Desigur, paginile pot fi de dimensiuni diferite, colorate etc.

Figura 17-33: Pagină a formularului

- Proiectați un formular pentru introducerea unor noi cititori până când se închide fereastra formularului. Se vor folosi obiecte de control diferite. Codul noului cititor se va realiza prin program, fiind chiar poziția în cadrulu fișierului.
- Realizați formularul următor, care să permită editarea simultană a tabelor Carti și operații. Fixați opțiuni diferite de adăugare a datelor.

Isbn	Codcit	Data_i	Data_r
1	22	08/09/99	08/09/99
1	34	08/10/99	/ /

Figura 17-34: Formularul de editare simultană

- Construiți un formular care să selecteze dintr-o listă cartea împrumutată, din altă listă numele cititorului și să adauge un articol în tabela Operații știind că data împrumutului este chiar data curentă.
- Construiți un formular care să țină controlul operațiilor de editare cărți, editare operații, editare cititori, sub forma unui **panou de bord** cu butoane de control care să dirijeze activitățile. Plasați o imagine de fundal.

Figura 17-35: Formular sub forma unui panou de bord

IV. Sarcini pentru construirea unui formular de actualizare simultană a mai multor tabele

Informațiile despre personalul unei societăți se țin în 3 fișiere:

PERSOANE (cod N(3), nume C(10), adresă M, studii M, stareciv C(1), buletin C(10));

ANGAJAȚI (cod N(3), loc_muncă C(10), funcția C(5), data_angaj D, mod_angajare C(5), data_plecare D, motiv_plecare M);

SALARII (cod N(3), salar N(7), avans N(7), rețineri N(7)).

Proiectați baza de date și un formular de introducere a datelor în momentul angajării unei persoane.

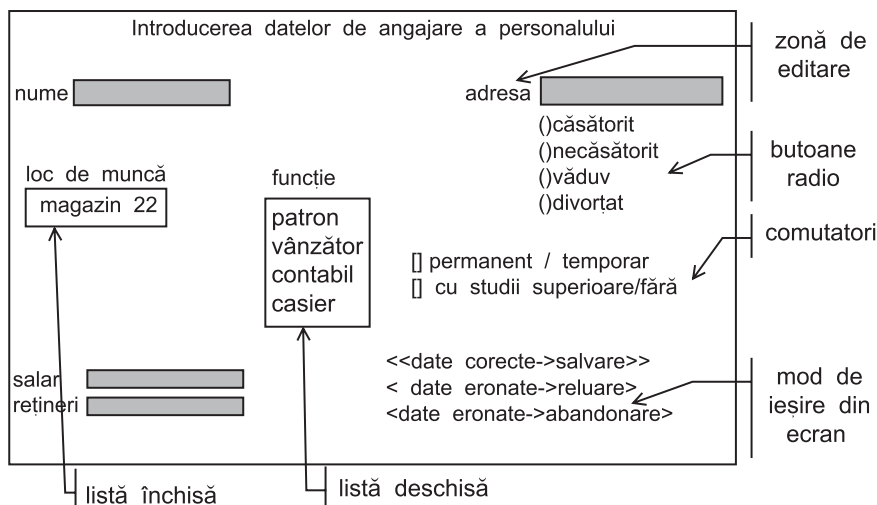


Figura 17-36: Schema de realizare a formularului

Țineți seama de următoarele cerințe:

- numele angajatului se va trece cu minuscule;
- data-angajării va fi chiar data înregistrării angajatului;
- mod-angajare va fi „salariat“, „prestări servicii“ codificat prin „SAL“, „PS“ dacă loc-muncă este „tonetă“ atunci funcția poate fi „vânzător“
- dacă funcția este „contabil“, nu trebuie să fie completat câmpul loc-muncă, iar modul de angajare va fi „salariat“.

Sarcini pentru realizarea unui mini-proiect

Pentru fiecare tabelă din baza de date proiectați formularele de vizualizare și actualizare. Folosiți atât Quick Form, cât și Form Wizard.

Proiectați pas cu pas un formular mai complex, care să conțină toate obiectele de interfață învățate.

Proiectați un formular cu rol de panou de bord al operațiilor realizate de aplicația voastră.

Afișarea datelor sub formă de rapoarte

- *Rapoarte. Generatorul de rapoarte Report Designer*
- *Etichete. Generatorul de etichete Label Designer*
- *Grafice. Utilitarul GENGRAPH*

Rapoarte

Rapoartele sau situațiile finale constituie o modalitate de valorificare a conținutului informațional al bazei de date, servind ca instrument de cunoaștere a aspectelor pozitive (și negative) dintr-o situație pentru care am cules date, le-am transportat, le-am verificat și depozitat într-o bază de date. Ținând seama de aceste aspecte, raportul apare ca un instrument al cunoașterii, care conține o cantitate mare de informații selectate, prelucrate, sistematizate după anumite reguli de prezentare, conform cerințelor de informare ale utilizatorilor.

În practică se disting mai multe tipuri de rapoarte¹:

- după aria de cuprindere și locul de obținere, rapoartele pot fi de uz local (pe secții, compartimente) sau de uz global (pe societăți, firme);
- după gradul de prelucrare a datelor, rapoartele pot fi analitice sau sintetice;
- după perioadele de referință a conținutului informațional, rapoartele pot fi zilnice, decadale, lunare etc.;
- după natura informațiilor oferite, rapoartele pot fi sub formă tabelară (text sau numeric) sau grafică (histograme, grafice).

Editarea unui raport presupune o anumită succesiune de operații:

1. Definirea conținutului informațional al raportului (se stabilesc datele care vor compune raportul, fișierele care le conțin, denumirea și ordinea câmpurilor din structura unei înregistrări, din liniile raportului).
2. Colectarea datelor care compun raportul (operații de selecție, interclasare, ordonare, filtrarea datelor care vor compune raportul).
3. Efectuarea calculului matematice (dacă este cazul, pentru rapoartele care conțin indicatori, aceștia sunt calculați și memorați în variabile sau câmpuri ale tabelor).
4. Definirea machetei raportului (operație complexă de definire a formei de prezentare și explicitare a informațiilor într-un cadru limitat – pagina sau ecranul – ținând seama de cerințele utilizatorilor).

¹ După Ion Lungu, „Sistemul FoxPro“, editura All, 1996.

Principalele categorii de informații sunt:

- datele propriu-zise ale raportului;
- indicatorii totalizați pe diverse niveluri de centralizare;
- capul de tabel;
- titlul raportului;
- datele sau nivelurile de grupare și totalizare;
- informații centralizatoare de sfârșit de raport/pagină.

5. Stabilirea modalităților de afișare (pe ecran, la imprimantă, într-un fișier).

Rapoartele pot fi realizate prin comenzi adecvate în programe utilizator, dar mai util este un program de generare a rapoartelor conform proiectului utilizatorului, cum este cel numit Report Designer.

Editorul de rapoarte Report Designer

SGBD-ul FoxPro pune la dispoziția utilizatorilor un program special, numit *generator de rapoarte*, prin intermediul căruia pot fi proiectate rapoartele conform cerințelor utilizatorilor. Informațiile despre proiect sunt salvate într-un fișier care, lansat în execuție, va afișa datele sursă (tabele/vederi), conform cu indicațiile proiectului.

Generatorul de rapoarte permite:

1. Proiectarea unui **raport simplu** (opțiunea **QUICK REPORT**), în care apare data curentă a listării, capul de tabel fiind format din denumirea câmpurilor din baza de date, iar conținutul – din articolele bazei.
2. Proiectarea unui **raport complex** prin indicarea în cadrul unor benzi speciale a componentelor raportului.
3. Introducerea benzilor de grup, pentru rapoarte care grupează datele având un câmp comun, fiecare grup fiind identificat prin antet, conținut și informații de final.
4. Folosirea funcțiilor standard (**recno()**, **pagno()**, **date()**, **time()**, **sum()**, **avg()**, **max()**), a câmpurilor calculate, a câmpurilor din baza de date sau a fișierului contextual deschis anterior.

Apelarea editorului

```
CREATE/MODIFY REPORT <fis.frx>
```

Lansarea în execuție

Lansarea în execuție a raportului se face prin comanda:

```
REPORT FORM <fis.frx> [HEADING<sir>] [NOEJECT]
[SUMMARY] <domeniu> [FOR <cond>] [WHILE <cond>]
TO PRINTER/TO FILE <fis.txt>
```

Comanda **REPORT FORM** permite listarea fișierului/vederii deschise anterior comenzii sau a fișierelor contextuale proiectate odată cu macheta raportului. Clauza **HEADING** are ca efect afișarea unui antet care va fi listat la începutul fiecărei pagini. **EJECT** inhibă saltul la pagina nouă înaintea afișării raportului. **SUMMARY** inhibă afișarea rândului curent și are ca efect listarea numai a liniilor de totaluri.

Mediul de proiectare

În vederea facilitării proiectării vizuale, Report Designer vă pune la dispoziție o bară de meniuri, un meniu contextual, o bară de instrumente, fereastra de proiectare, ferestre de dialog.

- Fereastra de proiectare** are mai multe benzi, unde vor fi depuse categoriile de informații ale raportului

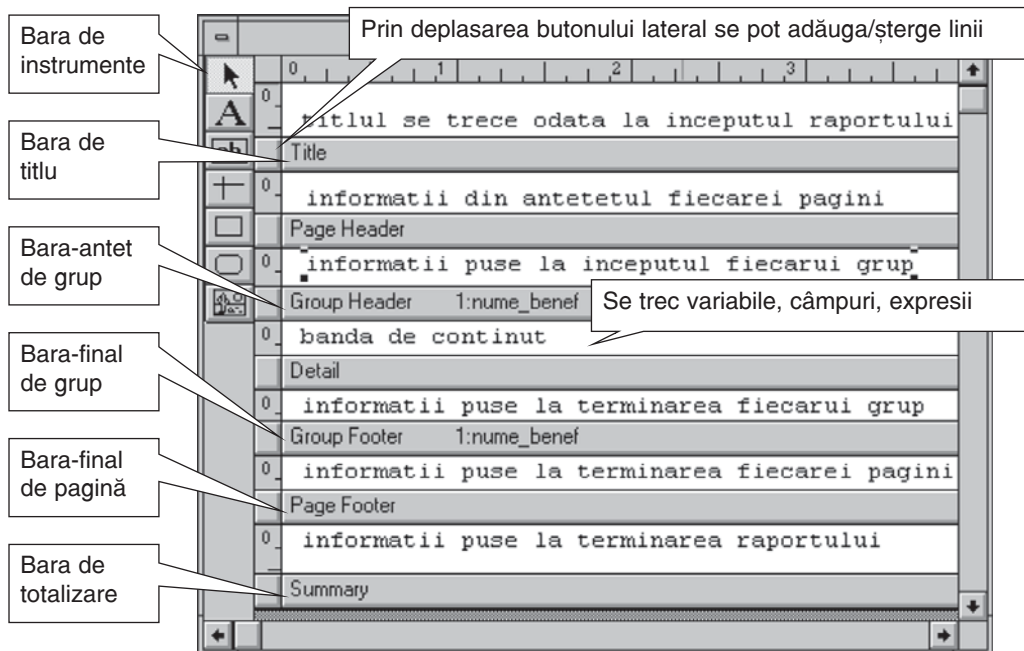


Figura 18-1: Fereastra de proiectare

Meniul Report

Vom explica particularitățile editării rapoartelor rezolvând pas cu pas o problemă.

Report	Window	Help
Title/Summary...	deschiderea benzilor de titlu și însumarea la final de raport	
Data Grouping...	deschiderea benzilor de grup	
Variables...	editarea variabilelor	
Default Font...	fixarea fontului implicit de raport	
Private Data Session	sursa de date este independentă de proiectul de raport	
Quick Report...	realizarea automată a unui raport rapid	
Run Report	execuția raportului	

Figura 18-2: Meniul Report

Tema 1. Folosirea Quick Report

Problema

Fie baza de date CONTRACTE (nr_contr N(3), data D, nume_benef C(10), nume_prod C(10), pret_promis N(5), cant N(5), data_livr D), care reține contractele comerciale a produselor unei firme. Se cere afișarea situației contractelor încheiate la data curentă sub forma unui raport:

LISTA CONTRACTELOR

nr_contr	data	nume_benef	pret_promis	cant
.....			

Rezolvare:

1. Apelăm generatorul de rapoarte cu comanda **CREATE REPORT** sau selectând **File, New, Report**.
2. Deschidem tabela CONTRACTE în mediul de date asociat raportului, selectând **Data Environment** din meniul **View** sau din meniul contextual al raportului.
3. Selectăm **Report, Quick Report**; se deschide fereastra Field Layout, ce permite alegerea formei și a conținutului raportului (Fields).
4. Câmpurile care vor fi folosite pot fi selectate din fereastra Field Picker.
5. Plasăm o bandă de titlu selectând **Report, Title** și dăm titlul raportului folosind butonul **Text**.
6. Inversăm coloanele **Cant** cu **Pret_promis**.
7. Selectăm **File, Page Setup** și stabilim caracteristicile globale ale raportului

din fereastra care apare. Putem preciza numărul de coloane (Number), distanța până la prima coloană (Left Margins), lățimea coloanelor (Width) și distanța dintre ele (Spacing). Raportul poate fi structurat pe coloane sau pe linii prin clic pe butonul Print Order.

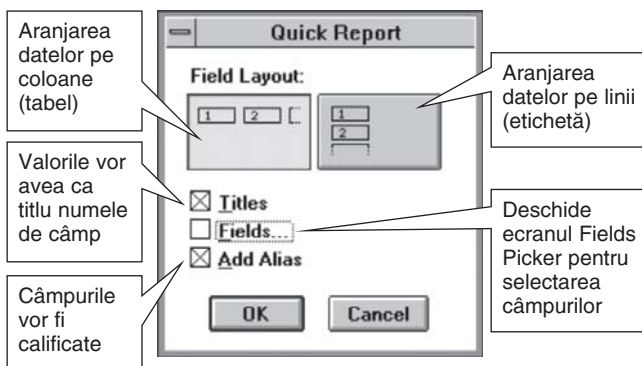


Figura 18-4: Fereastra Quick Report

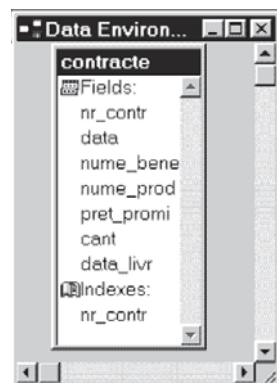


Figura 18-3: Tabela CONTRACTE în Data Environment

În general, imprimantele lasă o zonă la marginea hârtiei unde nu se afișează nimic (opțiunea Printable page). Dacă alegem Whole page, atunci întreaga pagină fizică este folosită pentru raport.

8. Raportul poate fi previzualizat selectând **View, Preview** sau folosind meniul contextual. Prin previzualizare observăm comportarea raportului pe setul de date. Observați bara de instrumente asociată. Revenim în modul de proiectare selectând **View, Design** sau prin închiderea ferestrei Preview.
9. Dacă suntem mulțumiți de funcționarea raportului, salvăm prin combinația **<CTRL>+<W>** sau selectând **File, Save R1.rfx**.

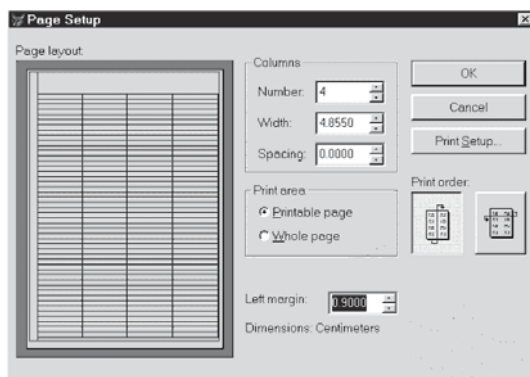


Figura 18-5: Fereastra Page Setup

10. Utilizăm raportul prin comanda: **REPORT FORM R1 FOR data=date()**

Tema 2. Proiectarea unui raport pas cu pas

Problema

Plecând de la tabela CONTRACTE (nr_contr N(3), data D, nume_benef C(10), nume_prod C(10), pret_promis N(5), cant N(5), clauze M) care reține contractele comerciale, să se proiecteze următorul raport:

Lista contractelor in perioada 08/08/99 - 08/08/99				
Nrc	Nr_contr	Beneficiar	Valoare	Clauze
1	121	Sc Intim	1000000	se va livra pana la 1 ian penalizare 5% /zi
2	122	Sc Intim	40000000	transport auto termen livrare : 1 decembrie 1999
3	123	Sc Alba	500000	transport CFR
4	124	sc Bacau	23000000	termen livrare : 1 ianuarie 2000 transport auto Plata □ fiecare joi
<i>Total valoare contracte</i>			64500000	

Observați fereastra de proiectare a raportului (fig. 18-6).

Proiectul trebuie să conțină ca titlu un *text* constant „Lista contractelor...” și intervalul de afișare dat din apel.

Numărul curent al contractului este obținut prin numărarea valorilor într-o *variabilă locală*; câmpurile contract, data, beneficiar și clauze sunt câmpuri din tabelă; valoarea fiecărui contract este o expresie (*câmp calculat*) obținută din produsul pret*cant. La sfârșitul raportului calculăm valoarea totală pentru contractele afișate. Obținem raportul numai pentru contractele produsului X într-un fișier text cu numele **Lista.txt**.

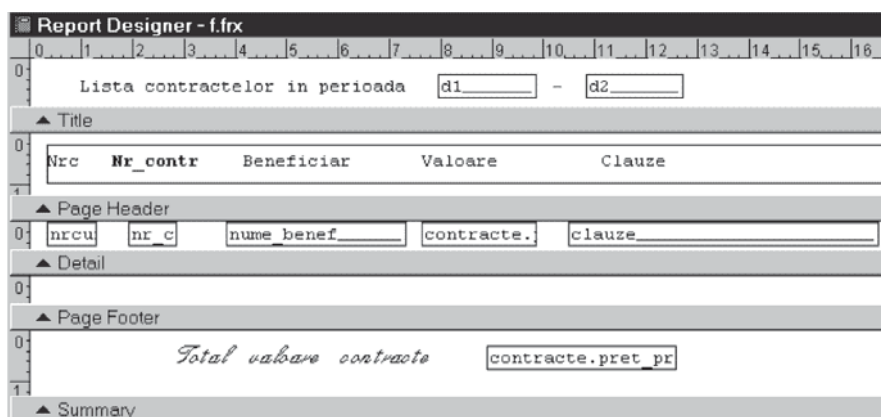


Figura 18-6: Fereastra de proiectare a raportului

Rezolvare:

1. Deschidem tabela Contracte.
2. Creăm variabilele globale D1 și D2 ca interval de selecție dorit; prin fereastra de comenzi li se atribuie valori.
3. Intrăm în generatorul de rapoarte prin **CREATE REPORT Z**.
4. Deschidem benzile de titlu și însumare (**Report, Title**) și plasăm textele dorite pentru titlu și cap de tabel (clic pe butonul **Text**).
5. Plasăm intervalul D1-D2 în fereastra de proiectare pe banda de titlu astfel:
 - a) Executăm clic pe butonul **Field**; se deschide fereastra **Report Expression**, în care introducem direct numele unei variabile sau deschidem fereastra Expression Builder.

Să comentăm puțin această fereastră: pe linia Expression putem introduce direct expresia care va fi afișată. Prin clic pe butonul alăturat se deschide Expression Builder, care permite construirea rapidă a expresiei. Poziția expresiei afișate în banda unde este definită poate fi precizată și prin butoanele radio: **Float** –

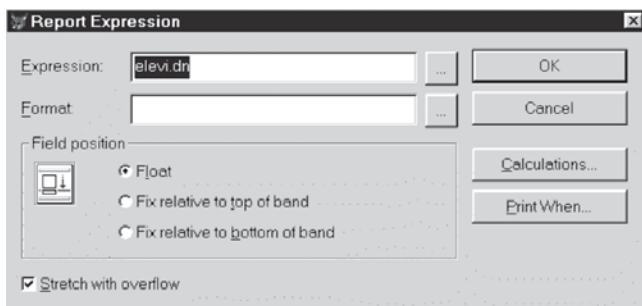


Figura 18-7: Fereastra Report Expression

– mobilă față de liniile deasupra ei²; **Fix relative to top** – fixă față de marginea superioară a benzii; **Fix relative to bottom** – fixă față de marginea de jos a benzii.

Opțiunea **Stretch with overflow** permite extinderea în jos a câmpului raportului cât este necesar pentru a cuprinde tot conținutul sursei de date.

² Este de obicei asociată unui câmp Memo și clauzei **stretch**. De exemplu, o adresă care poate avea mai multe linii deplasează obiectul plasat sub ea. Este o situație pe care o vom exemplifica ulterior.

- b) Selectăm variabila D1 din lista de variabile publice (creată la intrarea în proiectare; numele variabilei apare în lista utilitarului).
 - c) Plasăm linia separatoare prin butonul **Label**.
 - d) Plasăm variabila D2 în același mod ca și variabila D1.
5. Completăm bara **Detail** cu prima coloană – numărul curent al contractului, alegând una dintre variantele prezentate în continuare:
- a) Definim variabila Nr selectând **Report, Variable**; apare ecranul Variable Definition; dăm valoarea inițială zero și executăm clic pe butonul radio **(.)Count**. Comutatorul **Release After Report** face ca variabila să fie definită local raportului și să fie ștearsă la terminarea acestuia. Secțiunea Calculate conține principalele funcții care pot fi aplicate pentru calculul valorii variabilei respective în raport. Astfel, butonul **Count** permite ca variabila să conțină numărul de linii din tabelă, **sum** – suma valorilor expresiilor din definiția variabilei etc.

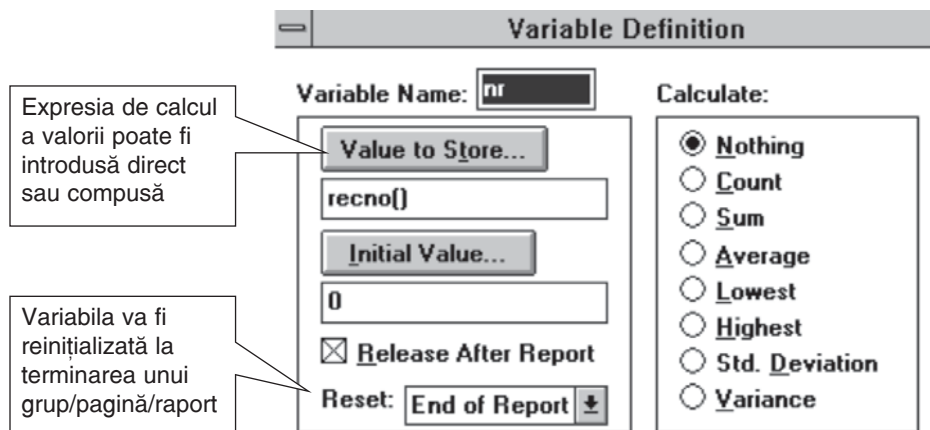


Figura 18-8: Fereastra Variable Definition

- b) Executăm clic pe butonul **Field** și alegem din Expression Builder un câmp numeric (de exemplu, Nr_contr). În fereastra Report Expression executăm clic pe butonul **Calculations**. Observăm aceleași funcții de calcul ca și în fereastra Variable. Alegem opțiunea **(.)Count**.
6. Plasăm câmpurile Data, Nume_benef, Clauze pe banda de detaliu prin aceleași manevre.
7. Stabilim formate corespunzătoare fiecărui câmp (ecranul Format).
8. Pentru proiectarea câmpului calculat Valoare din banda Detail care va afișa produsul între preț și cantitate, procedăm astfel: apăsăm butonul **Field**, iar în zona de editare a expresiei introducem produsul **Contracte.Pret_promis*contracte.Cant**. Putem folosi Expression Builder.
9. Parcurgem următorii pași pentru proiectarea câmpului calculat Valoare din banda Summary care va afișa suma valorilor contractelor:
- a) Deschidem fereastra pentru inserarea unui câmp (clic pe butonul **Field**, fereastra Report Expression) și introducem expresia: `contract.Pret_promis *contract.cant`
 - b) Afișăm fereastra Calculate și selectăm butonul radio **(.)Sum**, indicând operația de totalizare la nivel de raport a acestei expresii.

10. Salvăm (**File, Save**).

11.Executăm raportul printr-un program `p1.prg`.

P1.prg

```
accept 'dati data de inceput ' to d1
d1=ctod(d1)
accept 'dati data de sfirsit' to d2
d2=ctod(d2)
accept 'produs?' to x
report form z for nume_prod=x and between(data,d1,d2) to print
```

Putem seta condițiile de tipărire pentru fiecare câmp prin fereastra Print When.

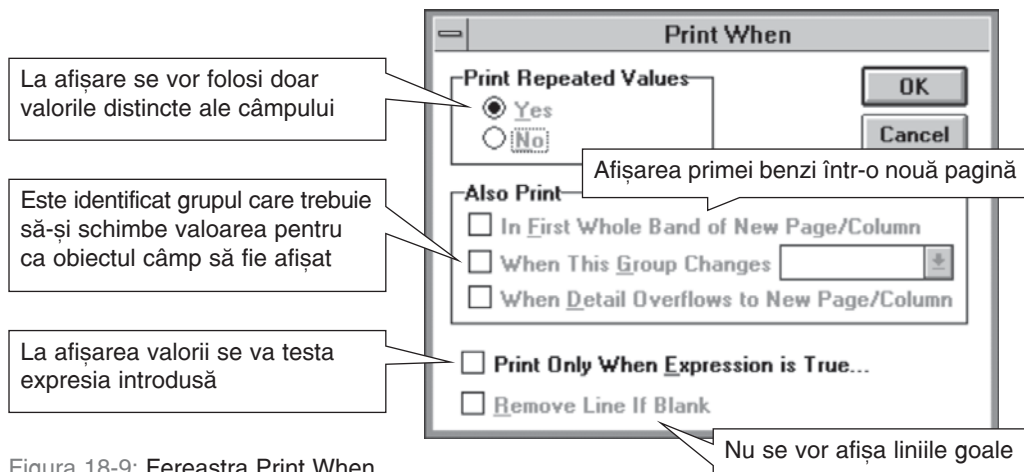


Figura 18-9: Fereastra Print When

Tema 3.Utilizarea benzilor de grup

Problema

Fie tabela CONTRACTE din temele anterioare. Se cere un raport care grupează toate contractele pe parteneri. Pentru fiecare partener vor fi afișate numărul, valoarea fiecărui contract, iar la sfârșitul listei – numărul de contracte și valoarea totală. Raportul va fi afișat în fișierul `Grup.txt`.

Indicații:

1. Deschidem baza de date CONTRACTE și indexăm după `nume_benef`.
2. Selectăm **Report, Data Grouping**. Introducem numele câmpului folosit ca și cheie de grupare în zona de editare sau construim prin **Expression Builder**.

SITUAȚIA CONTRACTELOR PE BENEFICIAR				
NUME BENEFICIAR		b1		
nrc	numar	contr	data	valoare
1	100	01.07.9		1000
2	101	02.07.9		500
TOTAL numar de contracte :				2
valoare contractata :				1500
NUME BENEFICIAR		b2		
nrc	numar	contr	data	valoare
3	103	09.09.9		2000
4	104	08.08.9		600
5	105	07.08.9		100
TOTAL numar de contracte :				3
valoare contractata :				2700

Callouts for the table:

- Beneficiary b1:** 'Pentru fiecare beneficiar va fi afișat numele (codul)'.
- Summary for b1:** 'Vor urma toate contractele sale numerotate, cu numărul, data și valoarea lor'.
- Beneficiary b2:** 'La finalul de grup va fi specificat numărul total de contacte și valoarea lor'.

3. Stabilim proprietățile grupului:

- **start group on new column** – este cazul rapoartelor multi-coloană, care încep o nouă coloană pentru o altă valoare a expresiei de grupare;
 - **start each group on a new page** – începe fiecare grup pe pagină nouă
 - **reset page number to 1 for each group** – numerotează paginile de la 1 pentru fiecare grup;
 - **reprint group header on each page** – tipărește antetul pe fiecare pagină a raportului, chiar dacă nu începe un nou grup;
 - **start group on new page when less than...** – fixează o distanță minimă față de marginea inferioară a raportului, astfel încât să nu înceapă un grup când nu este suficient spațiu pentru câteva linii măcar.
4. Revenim la ecranul de proiectare și completăm obiectele câmp pe benzile corespunzătoare; despre modul de proiectare a câmpurilor calculate am discutat deja la tema anterioară.
5. Previzualizăm (**Report, Preview**).
6. Salvăm (**File, Save**).
7. Rulăm cu comanda **Report form x**.

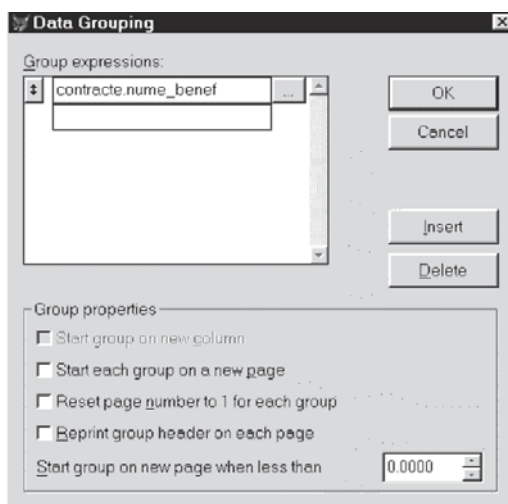


Figura 18-10: Fereastra Data Grouping

Tema 4. Folosirea rapoartelor pentru mai multe tabele

Problema

Baza de date despre contractele și facturile trimise beneficiarilor în contul acestor contracte conține tabela CONTRACTE.DBF (nr_contr N(5), nume_benef C(10), data D, nume_prod c(10), pret_promis N(7), cant N(5), clauze M) și FACTURI.DBF (nr_fact N(5), data D, nr_contr N(5), nume_benef C(10), nume_prod C(10), cant N(5), pret N(7)). Legătura dintre cele două tabele este realizată prin Nr_contr (numărul contractului).

Să se realizeze o situație a facturilor pentru fiecare contract în parte. Facturile sunt afișate cu numărul, data și valoarea fiecăreia. La terminarea listării facturilor unui contract se va afișa totalul valorii facturate.

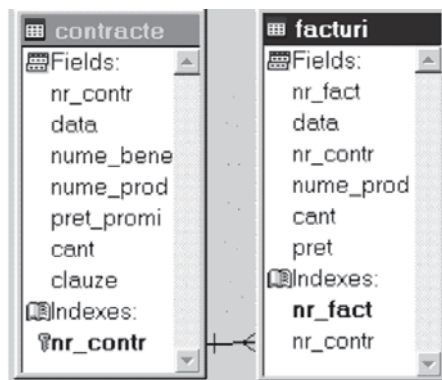


Figura 18-11: Tabelele CONTRACTE și FACTURI

Indicații:

1. Deschidem tabelele (izolate) în baza de date Contracte și stabilim o legătură 1-n între Contracte și Facturi.
2. Intrăm în proiectare prin **Create Report** sau selectând **File, New, Report**.

lista facturilor pe contracte

contractul	122	din	08/08/99	cu	Sc Intim
factura	data	produs	valoare		
1000	08/08/99	papuci	1000000		
1002	08/11/99	papuci	1000000		
1003	08/11/99	papuci	2500000		
total valoare facturata			4500000		

contractul	123	din	08/07/99	cu	Sc Alba
factura	data	produs	valoare		
1008	08/10/99	sandale	30000000		
total valoare facturata			30000000		

3. Deschidem Data Environment și adăugăm tabelele Contracte și Facturi. Observăm că legătura este 1-1. Pentru a o transforma într-o legătură 1-n edităm legătura (dublu clic pe legătură în fereastra Data Environment) și modificăm în ecranul Properties atributul **One-to Many = .T**.
4. Deschidem benzile de titlu și însumare (**Report, Title**).
5. Plasăm informațiile pe benzi, alegând fișierul corespunzător.

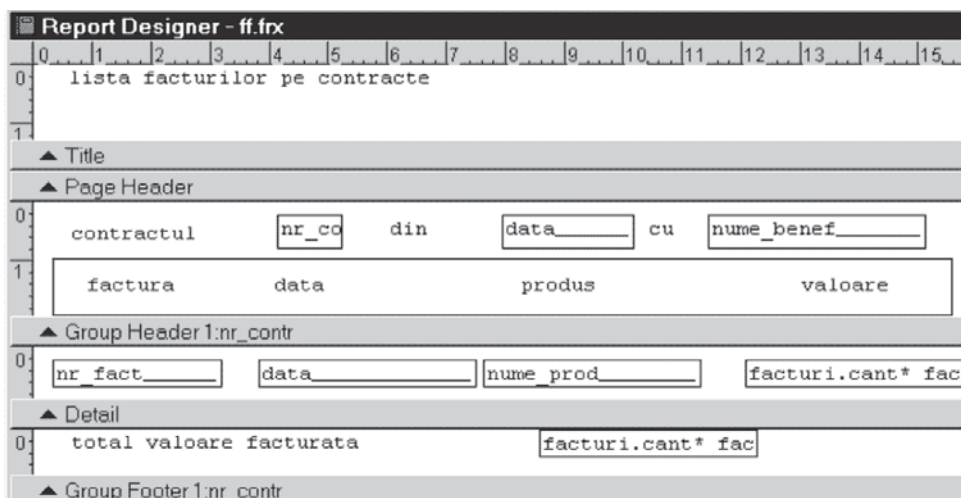


Figura 18-12: Proiectarea raportului

Salvăm și lansăm în execuție prin **Run, Report**.

Observație. Putem stabili contextul de lucru la proiectare prin fereastră, stabilind ca opțiunea **Report, Private Data Session** să fie activă. Aceeași secvență de stabilire a legăturii va fi plasată și în procedura de apel a raportului.

```
Use contracte in 1
Use facturi in 2
Select contracte
Set relation to nr_contr into facturi
Set skip to facturi
Report form ff
```

Tema 5. Folosirea raportului cu funcții utilizator

Problema

Pornind de la baza de date Contracte din exemplele anterioare se cere un raport sinteză care afișează în stânga datele despre contract, iar în dreapta informațiile despre facturile contractului, sub forma următoare:

contracte		facturi		
122	/ 08/08/99	nr-fact	data	valoare
Beneficiar:	Intim	1000	08/08/99	1000000
termen livrare :		1008	08/10/99	30000000
1 decembrie 1999		1002	08/11/99	1000000
		1003	08/11/99	2500000

Va fi folosită **o funcție utilizator** care returnează pentru fiecare nr_contr un șir cu informații despre facturile acestuia. Astfel, în banda de detaliu, vor fi trecute atât informațiile despre contracte (partea din stânga), cât și despre facturile lor (partea din dreapta). Pentru că variabila are lungime diferită, se va selecta clauza Stretch.

Pași:

1. Deschidem baza de date în Data Environment. Ambele tabele sunt indexate după Nr_contr.
2. Deschidem ecranul de proiectare a raportului și grupăm după Nr_contr. Observați ecranul de proiectare (figura 18-13).

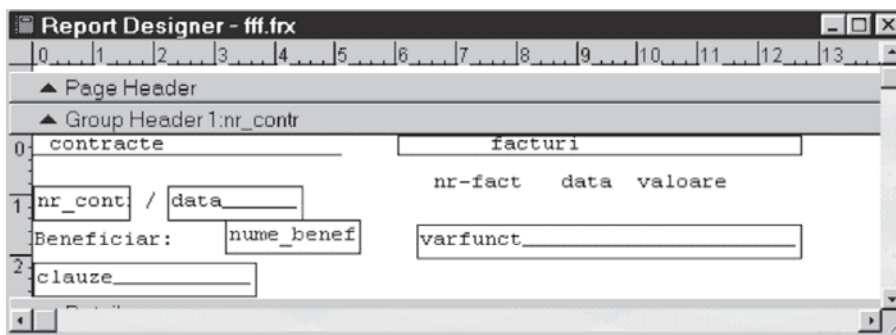


Figura 18-13: Proiectarea raportului

3. Pe banda de detaliu la nivelul grupului trecem atât textele explicative, cât și câmpurile din fișierul contracte: Nr_contr, data, nume_benef, clauze (care va avea comutatorul Stretch);
4. Creăm variabila m.varfunct de tip C, cu valoare inițială șirul vid și cu valoare memorată o funcție utilizator. Exemplu: **Store value=contr()**.
5. Plasăm variabila m.varfunct în stânga zonei proiectate și fixăm comutatorul **Stretch** pentru aceasta.
6. Salvăm raportul.
7. Scriem programul de apel care va avea și funcția utilizator **contr()**.

Programul apel_fff

conține funcția necesară raportului. Se execută programul cu DO apel_fff

Apel_fff.prg
report form fff
return

```
function contr
cvar=''
select facturi
seek facturi.nr_contr
scan while facturi.nr_contr=contracte.nr_contr
cvar=cvar+str(nr_fact,5)+' '+ dtoc(data)+
str(pret*cant)+
chr(13)+chr(10)    && trecerea la un nou rand
endscan
select contracte
return cva
```

Aplicația Report Wizard

Report Wizard este folosit pentru rapoarte simple, caz în care asistentul va cere doar fișierele și câmpurile pe care doriți să le folosiți. Există mai multe forme de rapoarte care pot fi realizate cu aplicația wizard: rapoarte simple pe baza unui singur fișier sau rapoarte cu date din mai multe fișiere legate dintr-o bază de date.

Aplicația Report Wizard se apelează selectând **Tools, Wizard, Report**.

One-to-Many Report Wizard



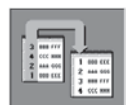
Pasul 1. Deschidem tabela părinte și selectăm câmpurile care vor forma raportul. De exemplu, dorim o situație a contractelor care să aibă asociate toate facturile acestora pe fiecare contract în parte. Selectăm câmpurile tabelii Contracte.



Pasul 2. Deschidem tabela copil și alegem câmpurile care vor fi trecute în raport. În exemplul nostru, folosim tabela Facturi.



Pasul 3. Stabilim cheia de legătură între tabele. În cazul nostru scriem: **Contracte.nr_contr= facturi.nr_contr**.



Pasul 4. Fixăm câmpurile de ordonare. În cazul nostru punem data contractului.



Pasul 5. Salvăm ca fișier **.QPR**, pe care îl putem rula imediat (opțiunea **Save and Run**) sau putem intra în Query Designer (opțiunea **Save query and modify it in Query Designer**). La fiecare pas pot fi previzualizate rezultatele.

Ultima fereastră permite alegerea formei de finalizare. Astfel, raportul poate fi previzualizat pe ecran (Preview), i poate da un titlu (Enter a title for your report), poate fi salvat pentru folosire ulterioară (Save report for later use), poate fi salvat odată cu afișarea (Save and print report) sau se poate deschide fereastra de proiectare a raportului pentru a i se face corecții.

Etichete

Etichetele (labels) sunt o altă modalitate de afișare a datelor. Etichetele pot avea declarată o dimensiune, ca număr de linii și coloane, pot fi afișate câte una sau mai multe pe lățimea hârtiei.

Să ne gândim la necesitatea scrierii unor etichete pentru medicamente, colete postale, cărți de vizită. Tot etichete pot fi „fluturașii“ cu informațiile despre salariul fiecărei persoane dintr-o unitate. Informațiile nume-persoană, funcție, salar_brut, rețineri, sporuri etc. vor fi însoțite de comentarii, încadrate în chenar și afișate pe mai multe coloane pentru a face economie de hârtie.

Această formă specială de raport poate fi proiectată prin intermediul utilitarului **Label Designer**. Utilitarul este asemănător cu Report Designer, folosește aceleași instrumente (meniu, fereastră de proiectare, bară de instrumente). Singura diferență este mărimea paginii folosite: dacă Report Designer setează o pagină întreagă, Label Designer își setează o mărime potrivită încadrării informațiilor în etichetă.

Apelarea generatorului de etichete se face prin comanda:

```
CREATE / MODIFY LABEL <fis.lbx>
```

Comanda lucrează în modul ecran și afișează un meniu care permite proiectarea unei etichete și salvarea proiectului într-un fișier special cu extensia **.Lbx**. Ecranul de proiectare este același cu cel afișat de generatorul de rapoarte.

Lansarea în execuție

```
LABEL FORM <fis.lbx> TO PRINTER / TO FILE <fis.txt>  
[<domeniu>] [FOR <cond>] [WHILE <cond>] [SAMPLE]
```

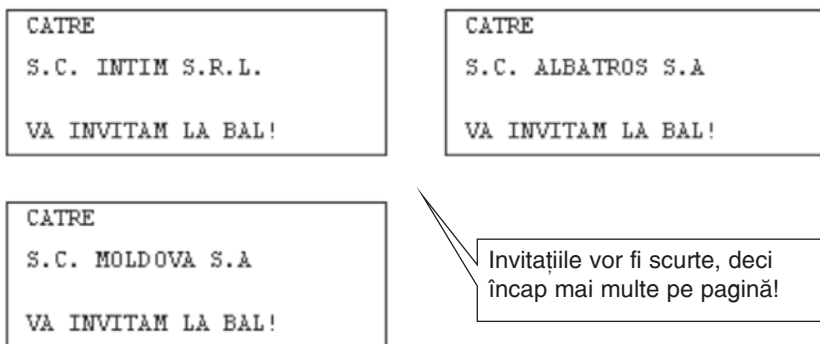
Comanda acționează pe articolele selectate prin clauzele **<domeniu>**, **FOR**, **WHILE**. Afișarea poate fi făcută pe ecran, sub forma etichetelor de probă (când este prezentă clauza **SAMPLE**), la imprimantă (**TO PRINTER**) sau într-un fișier text (**TO FILE <fis.txt>**).

Exemplu

Tema 6. Proiectarea unei etichete cu Label Designer

Problema

În vederea elaborării unor invitații, dorim afișarea numelui și adresei tuturor destinatarilor (beneficiarilor) sub forma unor etichete, ca în exemplul următor:



Rezolvare:

1. Deschidem fișierul Contracte și indexăm unic după Nume_benef.
2. Apelăm generatorul de etichete selectând **File, New (.)Label** sau prin comanda **Create Label** <nume-eticheta
3. Alegem forma de etichetă din formele predefinite afișate la deschiderea generatorului.
4. Introducem informațiile în banda de detaliu.
5. Previzualizăm prin **Report, Preview**.
6. Salvăm prin **File, Save**.
Executăm prin comanda **LABEL FORM** sau selectând **Run, Label**.

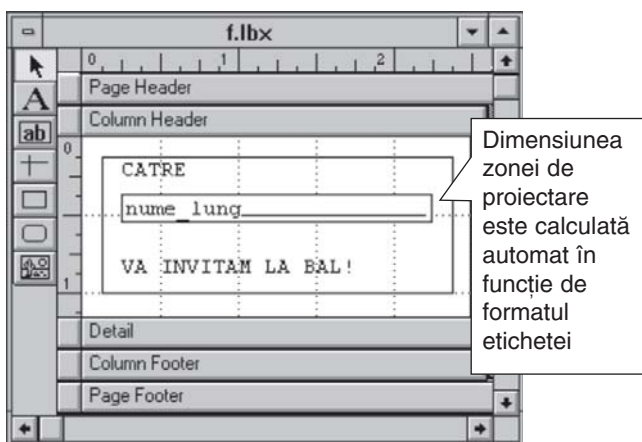


Figura 18-14: Proiectarea etichetei

Disponerea etichetelor pe foaia de afișare este realizată în funcție de configurarea din fereastra **Page setup**, prezentată la **Report Designer**.

Aplicația Label Wizard

Poate fi apelată selectând **Tools, Wizard, Label**; parcurgem ferestrele necesare proiectării etichetelor:

- Pasul 1: Selectarea tabelii sursă;
- Pasul 2: Selectarea formei etichetei din cele prestabilite;
- Pasul 3. Construirea machetei etichetei prin indicarea conținutului (texte și câmpuri din tabela selectată);
- Pasul 4. Ordonarea datelor;
- Pasul 5. Salvarea etichetei și/sau modificarea acesteia.

Fișierul creat are extensia **.lhx** și este utilizat prin comanda **LABEL FORM <fis.lhx>** sau selectând **Run, Label**.

Etichetele sunt definite linie cu linie. Fiecare câmp selectat este trimis în lista Selected Fields prin clic pe butonul Move „>”. Expresiile sunt construite întâi în zona de editare Text, apoi sunt mutate în Selected Fields. Nu uitați să includeți un spațiu separator între informații!

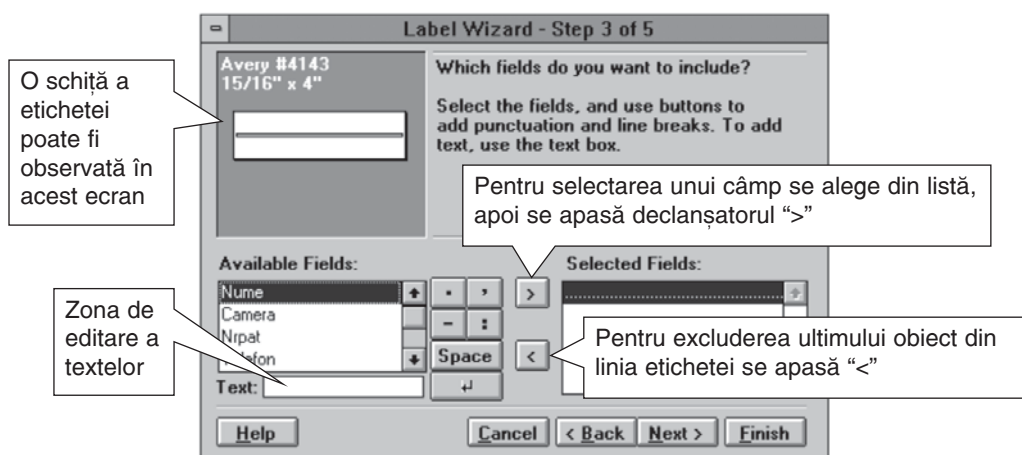


Figura 18-15: Folosirea aplicației Label Wizard

Grafice

Un alt tip de raport care poate fi realizat în mediul Windows prin intermediul sistemului FoxPro și pe baza conținutului tabelor gestionate de acesta, este raportul sub formă grafică. Dacă o balanță de plăți a beneficiarilor poate fi reprezentată sub forma unui tabel, fișele de cont ale clienților arată „bine” realizate cu Label Designer, un raport privind situația vânzărilor lunare pe magazine este mult mai sugestiv sub forma unui grafic. Desigur, o histogramă poate fi realizată printr-un program de grafică, dar pentru mulți dintre noi este o activitate destul de dificilă.

Utilitarul care realizează grafice este Graph Wizard, apelat selectând **Tools, Wizard, All, Graph**. Ca și la celelalte programe de asistență, parcurgând pașii indicați, poate fi realizat un grafic care va fi salvat într-un câmp de tip General, numit Olegraph, al unei tabeli numită implicit **vfpggraph.dbf** sau ca fișier **.scx** ce poate fi editat și executat prin **DO FORM**.

Atenție! Trebuie bine gândit ce grafic dorim să realizeze utilitarul, ce surse de date are la dispoziție, pentru că reprezentarea grafică nu se poate referi decât la câmpurile din tabela curentă.

Exemplu

Tema 7. Proiectarea unui raport de tip grafic

Problema

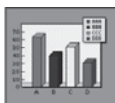
Pentru fișierul contracte dorim variația cantității contractate de beneficiari pentru produsul „Pantofi”.

Rezolvare:

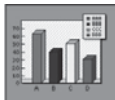
1. Folosim generatorul de interogări pentru a totaliza tabela Contracte pe câmpul nume_benef, cu filtrare pe produsul „pantofi”. Vom obține în coloana Cant totalul cantităților contractate pe beneficiari.
2. Deschidem utilitarul prin **Tools, Wizard, All, Graph**.



Pasul 1. Deschidem tabela de date pe care dorim să o folosim pentru grafic (Contracte). Fixăm câmpurile care vor fi trecute pe axe.



Pasul 2. Fixăm axele. Pentru o axă OY este necesar un câmp numeric și vom trece Cant. Pentru axa OX fixăm câmpul Nume_benef și executăm tragere și plasare până la fereastra din stânga.



Pasul 3. Alegem forma graficului.



Pasul 4. Salvăm graficul ca tabel sau edităm prin Browse.



sarcini de laborator

I. FOTBAL: Proiectați o bază de date pentru evidența echipelor, jucătorilor și meciurilor dintr-un campionat de fotbal. Populați baza de date cu date de test.

1. Scrieți comenzile SQL sau proiectați interactiv interogările pentru a afla:
 - a) cine face parte din echipa X;
 - b) programul competițional pentru cupa României;
 - c) unde joacă X (la ce echipă, de când..);
 - d) care sunt arbitrii pentru meciul x-y;
 - e) cu ce scor s-a încheiat meciul x-y;
 - f) care este ordinea echipelor după punctaj;
 - g) care este ordinea echipelor după golaveraj.

2. Proiectați formularele pentru vizualizarea și editarea datelor.
3. Proiectați următoarele rapoarte:
 - a) Pentru fiecare stadion pe care au loc meciuri, la planificarea acestora este necesar să fie trimisă câte o scrisoare cu următorul conținut. Proiectați-o prin Report Designer!

FEDERAȚIA ROMÂNĂ DE FOTBAL AMATOR	
Către administrația stadionului.....	
Avem deosebita plăcere să vă anunțăm planificarea meciului	
între.....și.....la data.....pe stadionul dumneavoastră.	
Rugăm asigurați buna desfășurare, eventual sponsorizare!	
azi,.....ora.....	Semnătura,

- b) Proiectați raportul final al campionatului.

Pentru fiecare echipă vor fi afișate meciurile jucate și scorul. La finalul raportului va fi scris totalul golurilor înscrise/primate de echipa respectivă, punctajul.

CAMPIONATUL NAȚIONAL DE FOTBAL			
Situație Finală			
Echipa.....			pag. x
data	loc	adversar	scor
total goluri înscrise.....		total goluri primate.....	
total golaveraj.....		total punctaj.....	

- c) Realizați un raport cu evidența cronologică a meciurilor pe stadioane. Pentru fiecare stadion va fi afișat punctajul maxim obținut de o echipă care a jucat în deplasare, respectiv acasă. La sfârșitul raportului se vor calcula numărul total de meciuri jucate și numărul total de goluri.

Lista meciurilor pe stadioane			
Stadionul:.....			
data	echipa1	echipa2	scor
punctaj maxim (rezultat meci) în deplasare:.....			
punctaj maxim (rezultat meci) acasă:.....			
număr de meciuri jucate pe stadion:.....			

4. Proiectați un formular tip prezentare a aplicației cu butoane care să deschidă formularele pentru editarea datelor și pentru rapoarte.



sarcini pentru realizarea unui mini-proiect

Fiecare aplicație va trebui să conțină diverse raportări. Proiectați rapoartele specifice aplicației, știind că trebuie să aveți:

1. un raport realizat pas cu pas cu date centralizate pentru o tabelă;
2. un raport cu date preluate din mai multe tabele;
3. un raport care folosește funcții utilizator;
4. un raport realizat cu Report Wizard;
5. un raport de tip etichetă;
6. un raport de tip grafic.

Proiectarea meniurilor

■ Generatorul de meniuri Menu Builder

Meniul este un element de interfață indispensabil unui proiect informatic profesional. Este un ansamblu de opțiuni pus la dispoziția utilizatorului, prin care se exercită controlul asupra aplicației. Meniul principal este plasat pe prima linie a ecranului aplicației, peste sau în completarea barei de meniuri. Este recomandabil ca orice operație să poată fi declanșată de utilizator prin alegerea ei dintre elementele meniului. Meniurile pot fi de mai multe tipuri: grafice sau în mod text, orizontale sau verticale.

Generatorul de meniuri Menu Builder

Crearea meniurilor utilizator este mai ușoară și mai rapidă prin utilitarul numit **Menu Builder**.

Apelul generatorului se face selectând **File, New (.)Menu** sau prin comanda:

```
CREATE /MODIFY MENU <fis.mnx>
```

Există două tipuri de meniuri care pot fi create cu acest utilitar: meniuri utilizator clasice, așezate pe linia zero, în locul sau în completarea barei de meniuri (tip menu) și meniuri scurtătură, de dimensiuni mici, afișate prin clic dreapta, plasate în fereastra utilizatorului.

Pentru crearea meniului, proiectantul va indica numele opțiunilor, operațiile declanșate de alegerea acestora, precum și caracteristicile globale ale mediului. Pe baza specificărilor făcute în fereastra de proiectare, este generat un program care prin lansare în execuție va activa meniul.

Programul obținut are extensia **.Mnx** și poate fi executat, ca orice program, prin **DO <fis.mnx>**.

La lansare, Menu Builder deschide o fereastră de proiectare și un meniu pe prima linie. Meniul Menu permite construirea rapidă a unui meniu asemănător celui standard (**Quick Menu**), inserarea opțiunilor (**Insert item**), inserarea submeniurilor (**Insert bar**), ștergerea opțiunilor (**Delete item**). Comanda **Generate** permite transformarea proiectului într-un fișier executabil. **Preview** oferă o previzualizare a proiectului înainte de finalizare.

Fereastra de proiectare a meniului cuprinde elementele definitorii pentru proiectarea meniului principal al aplicației.

Partea principală conține numele opțiunilor (**Prompt**), tipul acțiunii declanșate prin selectarea opțiunii (**Result**), care poate fi deschiderea unui submeniu, execuția unei comenzi sau a unei proceduri. Fiecare opțiune poate fi numită sau editată local în zona alăturată.

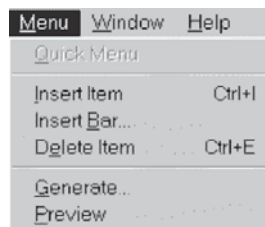


Figura 19-1:
Meniul Menu

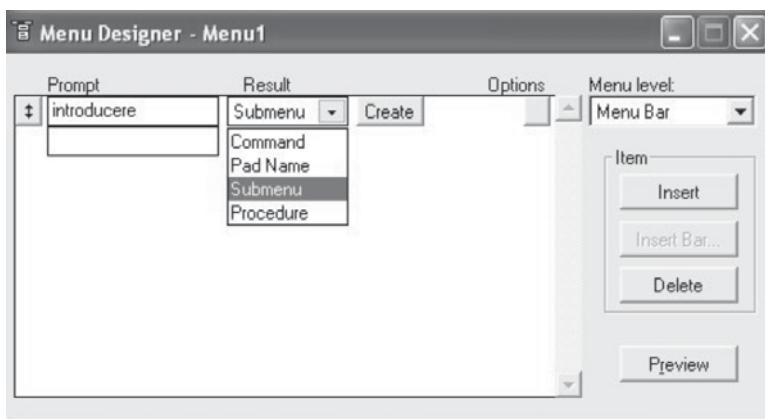


Figura 19-2: Fereastra de proiectare a meniului

Pentru opțiune se fixează câteva caracteristici (**Options**) cum ar fi:

- tasta de selecție rapidă în **Shortcut**;
- Condiția ca opțiunea să fie sărită (**Skip for**);
- Mesajul explicativ pe linia de stare la selectarea opțiunii (**message**);
- Alte comentarii pentru proiectant (**comment**).

Fixarea caracteristicilor globale ale meniului poate fi făcută prin fereastra **General Options**, deschisă selectând **View, General Options**.

Butoanele **Location** stabilesc poziția meniului relativ la linia zero a barei de meniuri. Astfel:

Replace – meniul utilizator înlocuiește bara de meniuri;

Append – meniul nou este adăugat la bara de meniuri;

Before/After – meniul nou este așezat înaintea/după submeniul indicat din lista derulantă alăturată, deschisă la activarea butoanelor respective.

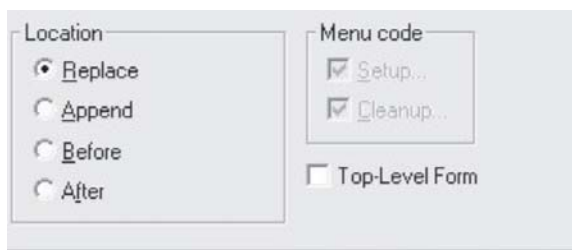


Figura 19-3: Opțiuni din fereastra General Options

Pentru un meniu se poate defini un cod de inițializare (**Setup**) care va fi executat la crearea meniului și de finalizare (**Cleanup**), executat la ștergerea meniului de pe ecran. Butonul **Edit** permite scrierea unei proceduri executate la activarea meniului.

Operația de generare a programului este apelată selectând **Menu, Generate**.

Odată generat, fișierul poate fi executat ca orice program, dar trebuie specificată extensia **.Mpr** (pentru că la comanda **DO** se consideră extensia implicită **.Prg**).

Bara de meniuri este disponibilă oricând dacă folosim comanda **SET SYSMENU AUTOMATIC**. Pentru activarea meniului Browse sau a altor operații, folosim tasta **F10** sau executăm dublu clic. Dacă există setarea **SET SYSMENU OFF**, atunci bara de meniuri este dezactivată.

Revenirea la afișarea meniului sistem Fox când terminăm operațiile din meniul utilizator se poate face prin **SET SYSMENU TO DEFAULT**.

Tema 1. Pașii proiectării unui meniu

Problema

Fie baza de date CONTRACTE cu informații despre contractele comerciale ale unei societăți comerciale și FACTURI cu evidența facturilor trimise beneficiarilor în contul contractelor încheiate. Structura celor două fișiere este: CONTRACTE (nr_contr N(3), data D, nume_benef C(10), nume_prod C(10), pret_promis N(5), cant N(5), data_livr D); FACTURI(Nr_fact N(5), nr_contr N(5), data D, nume_prod C(10), cant N(5), pret N(5)).

Se cere elaborarea unui program care grupează principalele operații legate de evidența contractelor și facturilor.

Un astfel de meniu poate fi cel alăturat:

Rezolvare

Introducere	!Actualizare	aFisare	iEsire
<u>C</u> ontrate		<u>F</u> acturi pe contrate	
<u>F</u> acturi		<u>C</u> ontrate fara facturi	

Pasul 1. Apelăm generatorul de meniuri prin comanda **CREATE MENU** sau selectând **File, New (.)Menu**. Se deschide fereastra de configurare a meniului principal (bară).

Pasul 2. Definim opțiunile meniului principal:

- tastăm textul în zona **Prompt** (și indicăm tasta rapidă) (Introducere, Actualizare, Afișare).
- alegem acțiunea dorită în cazul selectării opțiunii din lista închisă **Result** (submeniu, procedură sau comandă).

Prompt	Result	Options
\<Introducere	Submenu	Edit
\<<Actualizare	Submenu	Edit
a\<Fisare	Submenu	Edit

Figura 19-4: Definirea opțiunilor meniului

Executând clic pe butonul **Create** se deschide o fereastră în care pot fi definite opțiunile submeniului în același mod ca la meniul principal sau pot fi scrise comenzile procedurii apelate.

Prompt	Result	Options
\<Facturi pe contrate	Command	report form y preview ✓
\<Contracte fara factu.	Command	report form z preview

Figura 19-5: Definirea opțiunilor submeniului

Butonul **Edit** deschide ecranul pentru modificare. Pentru fiecare opțiune, definim submeniu dorit în același mod ca la meniul principal.

La ultima opțiune de ieșire din meniul bară se poate pune comanda **SET SYSMENU TO DEFAULT**. De asemenea, se poate scrie comanda **CLEAR EVENTS**, ce determină încheierea tratării evenimentelor.

Pasul 3. În fereastra afișată selectând **View, General Options** fixăm **SETUP** (procedurile de deschidere a fișierelor, asocierea indecșilor, stabilirea relațiilor) și **CLEANUP** (procedurile de închidere-ștergere a ecranului, închiderea fișierelor). Și în Cleanup se poate pune comanda **SET SYSMENU TO DEFAULT**.

Specificăm zona unde va fi plasat meniul creat (**View, General Options, Location** **(.)Replace**).

Pasul 4. Generăm programul (**Menu, Generate**) sub numele **Contracte.mpr**.

Pasul 5. Executăm programul generat prin comanda **do contracte.mpr**.



sarcini de laborator

1. Scrieți un program–meniu care permite planificarea meciurilor dintr-un campionat, înscrierea rezultatelor pentru toate meciurile dintr-o zi dată, afișarea clasamentului. (Punctajul se calculează astfel: 3 puncte pentru un meci câștigat, 1 punct pentru meci egal, zero puncte pentru un meci pierdut. Golaverajul reprezintă diferența între golurile date și cele primite.)
2. Realizați un meniu scurtătură, ca în figura 19-6.
3. Realizați un submeniu pentru activitatea de aprovizionare a unei unități economice, și anume determinarea necesarului de aprovizionare. Revedeți modalitățile de calcul la capitolul despre interogări.

Realizați un meniu ca în figura 19-7:



Figura 19-6:
Meniu scurtătură

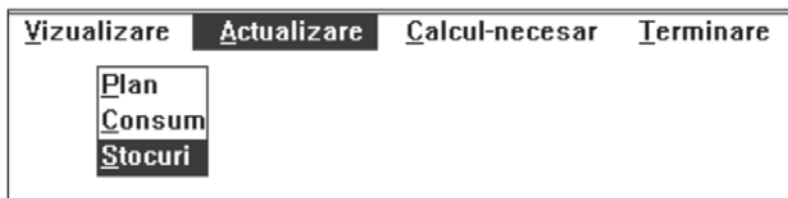


Figura 19-7: Meniul aplicației de aprovizionare

Opțiunea **Vizualizare** permite afișarea fișierelor (Read only).

Opțiunea **Actualizare** permite corectarea fișierelor: Plan, Stocuri, Consum.

Opțiunea **Calcul** determină și afișează necesarul de aprovizionat.

La **terminare**, se va reveni la bara de meniuri în FoxPro.

Proiecte și aplicații

- *Conducerea aplicației printr-un program monitor*
- *Organizarea aplicației sub formă de proiecte*
- *Generarea aplicațiilor executabile (.App, .Exe)*
- *Construirea dischetelor de distribuție*
- *Etape în realizarea unei aplicații informatice*

Conducerea aplicației printr-un program monitor

În general, o aplicație cuprinde o multitudine de programe, ecrane, rapoarte, meniuri, care au fost scrise pentru un scop comun și care sunt incluse într-o singură aplicație executabilă. Pentru buna funcționare a ansamblului este necesar ca toate operațiile pe care le face aplicația să poată fi lansate din opțiuni ale unui meniu.

De regulă, există un *program principal* sau *monitor* care deschide o fereastră sau o bară de meniuri, de care sunt legate toate celelalte operațiuni. Acest program mai are o sarcină importantă: aceea de a configura un mediu particular aplicației fără să deterioreze mediul găsit.

Observație: la realizarea unei aplicații informatice este foarte important să fie creat un mediu stabil și coerent, fără a face presupuneri legate de parametrii implicați. Configurarea existentă a mediului va fi salvată la intrarea în aplicație și refăcută la terminarea aplicației.

Mediul este configurat cu comenzi **SET**. Preluarea valorii curente a oricărui parametru stabilit cu comanda **SET** se face prin funcția **SET ()**.

Iată câteva aspecte de luat în considerare la proiectarea programului principal al aplicației:

1. Sunt salvate valorile curente ale parametrilor de mediu **SET**.
2. Sunt încărcate toate fișierele de proceduri și de clase, precum și bibliotecile API necesare.
3. Sunt deschise bazele de date.
4. Sunt configurați în mod corespunzător parametrii aplicației (mai ales cei referitori la tabele, **SET DELETED**, **SET EXCLUSIVE**, **SET MULTILOCK** etc.).
5. Este stabilită o rutină de tratare a erorilor cu comanda **ON ERROR**.
6. Este selectat și deschis fișierul de asistență.
7. Sunt stabilite și inițializate variabilele globale sau variabilele clasei aplicației.
8. Sunt activate meniul principal și bara de instrumente. Comanda **DO** apelează programul generat de Menu Builder ca fișier **.mpr**.
9. Este pornit procesorul de evenimente.
10. Sunt restabilitți parametrii de mediu modificați.

Exemple

Exemplul 1: În tema anterioară am proiectat un meniu al aplicației Contracte.mpr.

Pentru aplicația noastră ne propunem să scriem un program cu 3 funcții: una de salvare a parametrilor de mediu; alta de lansare a meniului aplicației și a treia de restaurare a parametrilor după terminarea aplicației.

Principal.prg

salvare parametri de mediu

```
OldEx=SET("exclusive")
Oldtalk= SET("talk")
Olddel=SET("Deleted")
Oldpath=SET('Path')
Olddir=FULLPATH(CURDIR())
```

Configurarea mediului

Do contracte.mpr

```
Read Events
```

Restabilirea mediului

```
Pop menu _Msystemenu
Set exclusive &OldEx
Set talk &Oldtalk
Set deleted &Olddel
! Cd (Olddir)
Set path to (Oldpath)
```

Programul generat de Menu Builder nu este un obiect, ci definește și afișează meniul aplicației, însă nu are propriul gestionar de evenimente. Comanda **READ EVENTS** activează sistemul de meniuri și, în plus, activează ciclul de procesare a evenimentelor. Ea continuă tratarea evenimentelor până în momentul lansării comenzii **CLEAR EVENTS**. Programul continuă cu linia de după **READ EVENTS**. Locul cel mai potrivit pentru plasarea comenzii **CLEAR EVENTS** este cel în care ați plasat opțiunea de terminare a meniului.

Organizarea aplicației sub formă de proiecte. Project Builder

Prin **proiect** FoxPro înțelegem un ansamblu de fișiere de proceduri, formulare, rapoarte, meniuri, care vor fi folosite în comun pentru crearea unei aplicații executabile.

Evidența acestui ansamblu de fișiere este dificilă, mai ales când numărul lor este mare.

De aceea, programatorii pot folosi utilitarul Project Builder în două moduri:

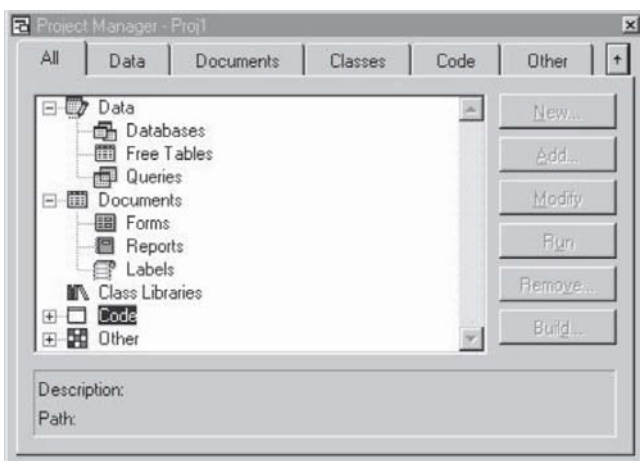


Figura 20-1: Fereastra Project Manager

1. Pentru sistemele informatice deja construite, utilitarul permite gestionarea fișierelor. Acest lucru presupune ca utilizatorul să includă manual fișierele în fereastra gestionarului. Este o operație destul de complicată, de aceea se recomandă folosirea Project Manager pentru organizarea aplicației de la zero, apoi pe măsură ce este construită.
2. Pentru sistemele informatice care urmează a fi construite, utilitarul este un adevărat centru de comandă al întregii activități de construire: de aici se pot lansa toate celelalte instrumente FoxPro pentru dezvoltarea aplicației.
3. Utilitarul Project Manager poate fi lansat selectând **File, New, Project** sau prin comanda **CREATE PROJECT**.

Operații cu fișierele din Project Manager

1. Un fișier sau element al proiectului poate fi adăugat după poziționarea pe subdirectorul corespunzător tipului fișierului și apăsarea butonului **Add**. Apare o fereastră pentru localizarea fișierului, iar acesta este plasat în subdirectorul proiectului.
2. Crearea unui nou fișier poate fi făcută după selectarea directorului corespunzător tipului său prin clic pe butonul **New**, care afișează fereastra de creare.

3. Ștergerea unui fișier poate fi făcută prin clic pe butonul **Remove**. Fișierul poate fi șters doar din proiect sau chiar de pe disc.

4. Putem să executăm un fișier-machetă sau un program prin clic pe butonul **Run**; astfel putem să (re)vedem rezultatele programului. La deplasarea peste directoarele cu rapoarte sau etichete, caseta de dialog pune în evidență butonul **Preview**, prin care putem previzualiza fișierul.

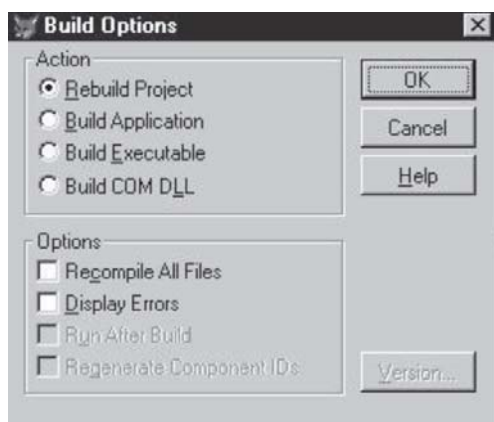


Figura 20-2: Fereastra Build Options

5. Modificarea unui fișier poate fi făcută prin clic pe butonul **Modify**, care apelează utilitarul Designer potrivit.
6. Recompilarea fișierelor este posibilă prin clic pe butonul **Build**, care deschide fereastra **Build Options**. Prima opțiune, **Rebuild Project**, înseamnă reconstruirea proiectului. Opțiunea **Recompile all files** permite ca la selectare să fie recompileate toate componentele proiectului, iar dacă rămâne neselectată determină recompilarea numai a elementelor care au suferit modificări față de data ultimei construiri a proiectului.

Observație. Un proiect este de fapt o tabelă cu o structură specială, având extensia **.Pjx**. Înregistrările corespund elementelor proiectului. Includerea unui element într-un proiect nu înseamnă includere fizică, ci doar memorarea informațiilor legate de fișier (poziție pe disc, data ultimei modificări etc.).

Generarea aplicațiilor executabile cu Project Manager

Project Manager permite ca, pe baza componentelor din arborele de directoare, să fie obținut automat un fișier executabil sau o aplicație. Acest lucru poate fi realizat prin clic pe butonul **Build**, care deschide fereastra Build Options, ce conține următoarele opțiuni:

- **ReBuild Project** – parcurge toate fișierele proiectului în vederea generării codului sursă și/sau a depistării erorilor.
- **Build Application** – parcurge toate fișierele proiectului pentru a construi o aplicație dependentă de mediul Fox, pe care o vom lansa cu comanda `DO <fis.App>`.
- **Build Executable** – generează un fișier executabil independent de mediul FoxPro. Build Executable generează proiectul și înglobează fișierele, la fel ca în cazul generării aplicației, dar adaugă modulele speciale Fox pentru rulare (Runtime), cât și bibliotecile și modulele necesare pentru fișierele **.Exe**.

La solicitarea de generare a unei aplicații are loc (re)compilarea automată a componentelor. Dacă procesul de compilare decurge normal, ultimul mesaj din bara de stare este *build application completed* (construirea aplicației s-a încheiat). Dacă selectăm **Display Errors**, erorile de compilare vor fi prezentate detaliat într-o fereastră, după compilare.

Atenție! În produsele executabile există unele comenzi sau opțiuni care nu sunt disponibile: **SET STEP**, **SET ECHO**, **BUILD PROJECT**, **MODIFY MENU/PROJECT**, **MODIFY QUERY**, **SUSPEND**, **CREATE VIEW**.

Ecranul **Project information** din meniul **Project** poate consemna diferite informații despre un proiect. În afara informațiilor privind autorul și asigură drepturile de autor – codul generat este individualizat cu ajutorul informațiilor referitoare la autor și dată.

Directorul Home este directorul rădăcină al proiectului. Caseta Debug info permite includerea sau excluderea numerelor de linie din codul sursă în fișierele compilate. Acest lucru este esențial la depanarea aplicației. Informațiile referitoare la programul și liniile care au generat eroarea rămân în fișierul de erori și permit corectarea.

- Caseta de validare Encrypt comunică programului FoxPro să cifreze fișierele compilate, asigurând protecția împotriva pirateriei software.
- Caseta Attach Icon permite asocierea unei pictograme proiectului.

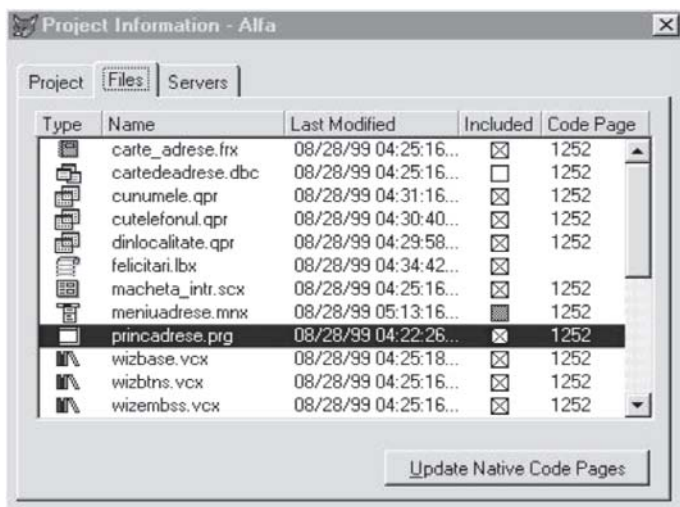


Figura 20-3: Fereastra Project Information

- Lista lungă de fișiere din tab-ul Files al ferestrei Project Information specifică informații despre toate fișierele conținute în proiect. Ordinea de afișare poate fi schimbată prin clic pe butoanele Type, Name și Last Modified.
- Caseta Included permite includerea sau neinclusiunea fișierului în aplicația .App sau .Exe.

Specificarea fișierului principal al aplicației

Orice aplicație trebuie să aibă un **fișier principal**. Toate celelalte fișiere sunt ancorate de acest fișier principal. Pentru specificarea acestuia, poziționăm cursorul pe fișierul dorit și selectăm **Project, Set Main**.

Alte facilități oferite de utilitarul Project Manager

1. Restrângerea ferestrei Project Manger la o bară de instrumente



Figura 20-4: Reducerea ferestrei Project Manager la o bară

Pentru că fereastra Project Manager este o excelentă modalitate de organizare a fișierelor aplicației, dar ocupă un spațiu destul de mare, sistemul Fox oferă o facilitate: restrângerea ferestrei la o bară de instrumente. Bara poate fi plasată pe primele linii ale ecranului printre celelalte butoane. Pentru a realiza transformarea ferestrei Project Manager într-o bară de instrumente, ne poziționăm pe titlul ferestrei și executăm clic pe butonul cu săgeată în sus. Toate funcționalitățile instrumentului Project Manager pot fi folosite prin clic dreapta.



Figura 20-5: Observați butonul ce permite reducerea ferestrei la o bară

2. Ancorarea unui tab al ferestrei Project Manager

De asemenea, un tab (de exemplu, de documente) poate fi ancorat într-un colț al ferestrei utilizatorului astfel: poziționăm cursorul pe un tab al ferestrei în forma sa de bară de instrumente, apoi tragem și plasăm către o altă zonă a ecranului. Observăm o fereastră ca în figură. Închiderea sa va determina revenirea tab-ului la bara de instrumente Project Manager.

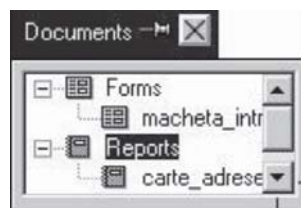


Figura 20-6: Tab-ul ancorat

3. Funcționarea ferestrei Project Manager ca meniu utilizator

Când este strâns și ancorat, instrumentul Project Manager poate fi folosit ca meniu al unei aplicații. Ca orice meniu utilizator, după poziționarea mouse-ului pe un fișier, acesta trebuie să fie executat sau deschis pentru previzionare. Urmați pașii:

1. Alegeți **Tools, Options, Projects**;
2. Din fereastra ce apare selectați butonul radio **Run** din grupul **Project double-click action**.

Realizarea dischetelor de distribuție

După ce am realizat o aplicație, este necesar să ne punem problema distribuirii ei către client. Transportul aplicației se face pe dischete (se pot folosi și CD-uri!) într-un format compactat. Dischetele de distribuție trebuie să conțină toate componentele necesare rulării aplicației. Dacă am realizat fișere **.app** este necesar să fie în prealabil instalat mediul FoxPro. Dacă am realizat fișere **.Exe** nu mai este necesară instalarea Visual FoxPro pentru execuție.

Oricum, ceea ce trebuie să realizăm este gruparea fișierelor aplicației pe dischete și oferirea unui program care să facă instalarea programului la destinatar.

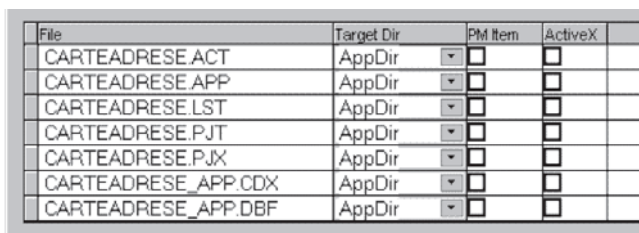
Utilitarul de construire a dischetelor de distribuție Setup Wizard

Să pornim de la exemplul anterior. Am realizat aplicația **alfa.app**. Toate fișierele se găsesc pe directorul **C:\ALFA**. Ne propunem să realizăm programul de instalare și dischetele de distribuție.

Pasul 1: Selectăm directorul în care se găsește aplicația pe care dorim să o trecem pe dischete. Este imperios necesar ca toate fișierele proiectului din care a fost generată aplicația (nu numai fișierul **.App** sau **.Exe**) să se afle într-un director separat. În cazul nostru, folosim **c:\ALFA**.

Pasul 2: Selectăm din lista afișată componentele speciale folosite de aplicație: Visual FoxPro Runtime este necesar când distribuim aplicația sub formă executabilă. Dacă aplicația folosește grafice, bifăm opțiunea Microsoft Graph. Opțiunea ODBC Drivers... va fi bifată dacă în sistemul informatic distribuit există conexiuni către baze de date externe mediului FoxPro.

Pasul 3: Specificăm un director pe discul de lucru – imagine a dischetelor care vor fi construite de generator. De asemenea, specificăm formatul dischetelor de distribuție (3,5 inch, 1,44 Mb). Opțiunea Web Setup este folosită la crearea unui director care conține toate fișierele în formă comprimată, pentru distribuirea aplicației pe internet. Net Setup este folosită în cazul instalării în rețea, când kit-ul de instalare nu este împărțit din segmente. În cazul nostru vom bifa opțiunea 1.44Mb.



File	Target Dir	PM Item	ActiveX
CARTEADRESE.ACT	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE.APP	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE.LST	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE.PJT	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE.PJX	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE_APP.CDX	AppDir	<input type="checkbox"/>	<input type="checkbox"/>
CARTEADRESE_APP.DBF	AppDir	<input type="checkbox"/>	<input type="checkbox"/>

Figura 20-7: Specificarea fișierelor

Pasul 4: Introducem titlul aplicației în caseta de dialog Setup Dialog Box Caption. Numele autorilor și informațiile de copyright vor fi introduse în Copyright Information.

Post Setup Executable permite introducerea numelui unui program care va fi executat la sfârșitul instalării aplicației.

Pasul 5: Precizăm directorul implicit care va fi creat pe discul de destinație, în care vor fi plasate fișierele aplicației la instalare, Default Directory. Dacă dorim adăugarea aplicației în meniul Start al sistemului de operare Windows sau, în general, pentru crearea unui grup de pictograme ale aplicației, folosim Program Group.

Pasul 6. Afișăm o tabelă cu fișierele aplicației. Pentru fiecare poate fi specificat în coloana Target Dir dacă fișierul urmează să fie instalat în directorul aplicației (AppDir) sau în directorul Windows (WinDir) sau în directorul Windows/System (WinSysDir).

În vederea specificării unor proprietăți ale fișierului, cum ar fi asocierea unei pictograme, se deschide fereastra Program Manager Item prin clic pe PM Item.

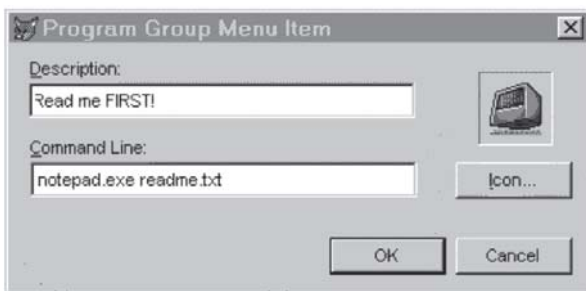


Figura 20-8: Fereastra Program Group Menu Item

Pasul 7: Înainte de începerea generării kit-ului, putem crea o arhivă autoextractivă în vederea distribuirii optime pe internet, prin opțiunea **Generate a web executable file**.

Pentru crearea dischetelor de instalare, Setup Wizard efectuează următoarele operații:

1. Verifică arborele de distribuție în căutarea fișierelor noi sau actualizate;
2. Actualizează propriul tabel de fișiere;
3. Comprimă toate fișierele;
4. Sparge fișierele în felii de dimensiunea unei dischete.

După generarea dischetelor, este afișată o statistică asupra numărului de dischete, spațiul ocupat de fișierele de pe fiecare dischetă etc.



sarcini de laborator

Liceul de Informatică are o bază de date necesară evidenței personalului și calculul salariilor. Considerăm următoarele tabele:

PERS (cod, nume, adresă, studii, data-nașterii, data-angajării, funcția, salar-brut, este-diriginte, specialitatea, catedra, grad didactic, alte-inf), care reține toate datele personale ale fiecărui angajat;

REȚINERI (cod-persoana, cod-reținere, suma, perioada-de-reținere), cu obligațiile personalului către terți; perioada de reținere poate fi dată ca interval de două date calendaristice;

SALARII (luna, cod-persoană, numar-ore-suplimentare, total-rețineri, total-sporuri, impozit, rest-plata), cu date despre salariile angajaților în fiecare lună a anului current.

Știm că:

- Vechimea este calculată în ani și se acordă un spor de vechime (mărit procentual cu 5% din salariu la fiecare 5 ani vechime).
- Ora suplimentară se plătește ca o oră obișnuită, dar este impozitată cu 10%.
- Salariul brut se consideră convenit pentru o normă de 18*4 ore/lună.
- Se acordă spor de pericolozitate tuturor cadrelor didactice care sunt diriginți (3% din salariul brut).
- Avansul chenzinal este 45% din salariul brut.

Realizați un program condus de un meniu principal pentru activitatea de administrare a salariilor, care să cuprindă următoarele activități:

- a. vizualizarea și editarea informațiilor legate de persoane, rețineri, salarii;
- b. afișarea listei avansului chenzinal ca o situație simplă cu numele și avansul convenit persoanei;
- c. afișarea raportului retribuțiilor sub forma următoare:

LISTA RETRIBUȚIILOR.....								
cod	nume	funcția	salar-brut	vechime	sporuri	rețineri	avans	rest-plată
total sume			xxxxxx		xxxx	xxxx	xxxx	xxxxx

- d. afișarea unor fluturași cu toate informațiile legate de plata salariului;
- e. afișarea graficului ponderii totalului reținerilor în total salarii brute pe unitate.



sarcini pentru realizarea unui mini-proiect

Integrați într-un program unitar componentele realizate pe parcursul lecțiilor anterioare. Construiți meniul și programul principal. Folosiți Project Manager și generați o aplicație executabilă. Generați dischetele de distribuție.

Dezvoltarea profesională în domeniul IT

- *Identificarea aptitudinilor pentru anumite tipuri de activități*
- *Crearea unui CV și reguli de susținere a unui interviu*
- *10 reguli în susținerea unei prelegeri*
- *Principii de lucru în echipă*

Domeniul IT este în plină expansiune în țara noastră, iar perspectivele profesionale pe care le oferă în alte țări sunt și ele interesante. Apariția continuă a noilor tehnologii, provocările intelectuale și satisfacțiile profesionale, dar și recompensele materiale pe care le primesc angajații din acest domeniu sunt printre factorii care determină mulți tineri să se orienteze către IT. Acest capitol vă prezintă pe scurt câteva sfaturi de care să țineți cont dacă vă gândiți să vă construiți o carieră în IT.

Identificarea aptitudinilor pentru anumite activități

Deși ați putea fi tentat de un post bine plătit de programator, de tester de inginer software, trebuie să vă gândiți mai întâi dacă sunteți potrivit pentru postul vizat, atât ca pregătire tehnică, dar și din punctul de vedere al calităților personale pe care le presupune. Iată câteva profesii din domeniul bazelor de date și calitățile necesare pentru ele:

Administrator de baze de date – este de obicei acea persoană care administrează instruirea datelor dintr-o firmă, controlând modificările structurilor de date, stocarea unor noi proceduri, menținerea și funcționarea optimă a bazelor de date.

Programator de aplicații pentru baze de date – poate că știți programare, dar aveți nevoie de experiență de lucru cu datele. Abilitățile de lucru cu bazele de date vă vor ajuta să atacați lumea aplicațiilor pentru baze de date, care au un impact direct asupra operațiunilor unor firme. Însă, ca să dezvoltați o aplicație pentru baze de date destinată unei firme, trebuie mai întâi să înțelegeți foarte bine întreaga activitate a firmei.

Analist pentru baze de date – poate nu doriți să dezvoltați programe, ci sunteți mai degrabă o persoană care înțelege bine firma și datele legate de aceasta. Un post cheie în acest amalgam este cel de analist al bazei de date. Analistii sunt planificatori cheie în designul unei baze de date; să nu uităm că proiectarea și dezvoltarea aplicațiilor sunt diferite, iar primul pas este proiectarea.

Crearea unui CV și reguli de susținere a unui interviu

După ce ați stabilit zona profesională care v-ar interesa și v-ați dat seama că aveți calitățile tehnice și personale pentru un anumit post, cum veți face să îl obțineți? La solicitarea oricărui post de muncă, angajatorul vă va solicita un CV (curriculum vitae). Acest document are în general o structură-tip, urmând anumite șabloane, iar rolul său cel mai important este de a convinge un potențial angajator că sunteți persoana potrivită pentru postul respectiv.

Structura conține câteva zone principale, pe care le completați cu datele proprii:

- Date personale: adresă, telefon, adresă e-mail, vârsta etc.
- Educație: atestă nivelul de cunoștințe pe care îl aveți, prin lista școlilor și a cursurilor urmate și a titlurilor pe care le-ați obținut în urma absolvirii lor.

- **Experiență profesională:** conține detalii despre locurile de muncă anterioare, responsabilitățile pe care le-ați avut în diferite posturi și succesele pe care le-ați obținut în muncă
- Diverse alte informații, ca aptitudini, pasiuni și interese personale, faptul că dețineți permis de conducere, aveți stagiul militar satisfăcut, nu aveți cazier etc.

Este bine să țineți cont că, în funcție de postul pe care îl solicitați, puteți schimba structura și conținutul CV-ului, pentru a scoate în evidență aspectele pe care le considerați mai relevante pentru compania și postul respectiv. După ce ați realizat CV-ul și ați „aplicat” pentru (ați solicitat) postul vizat, compania a făcut o selecție a candidaților și v-a convocat la un interviu. Cum trebuie să vă comportați la interviu pentru a face încă un pas spre obținerea postului? Iată câteva sfaturi:

- Mențineți o atitudine deschisă și pozitivă
- Răspundeți sincer și concis la întrebările care vi se pun
- Insistați asupra aspectelor bune, a calităților și succesorilor pe care le-ați înregistrat
- Informați-vă asupra companiei, a activității pe care o desfășoară și a succesorilor pe care le-a înregistrat
- Pregătiți întrebări despre companie, detalii despre post, colegi, mediul de lucru
- Îmbrăcați-vă adecvat domeniului și postului solicitat

10 reguli în susținerea unei prelegeri

Referatele și prezentările pe care le-ați susținut în cadrul orelor de la școală sunt un bun început pentru cele pe care le veți avea de susținut mai târziu, pe parcursul activității dumneavoastră profesionale. Iată câteva reguli care vă vor netezi drumul spre susținerea cu succes a unei prelegeri:

- Pregătiți conținutul prezentării
- Pregătiți sala: lumină, așezarea scaunelor, locația prezentatorului
- Pregătiți echipamentul: configurați computerul, rezoluția ecranului, dezactivați screen saver-ul, proiectorul
- Pregătiți notele
- Stabiliți măsuri de urgență pentru situații neprevăzute
- Planificați-vă mișcările în timpul prezentării
- Înțelegeți caracteristicile publicului
- Stabiliți strategii de captare a interesului publicului
- Stabiliți strategii de calmare a celor care întrerup și deranjează prezentarea (public „public ostil”)
- Faceți o repetiție înainte de susținerea propriu-zisă, cu ajutorul unui prieten

Principii de lucru în echipă

Domeniul IT este unul dintre domeniile în care munca în echipă este esențială. Produsele software sunt rezultatul muncii în echipă, iar eficiența echipelor este vizibilă și în succesul de piață al acestor produse. Un bun team player știe să asculte, să pună întrebări, să convingă, să respecte ideile altora, să ajute, să comunice, să participe la activitatea echipei.

Sarcini pentru realizarea unui mini-proiect

Organizați o echipă și stabiliți roluri: unii vor fi angajatori, ceilalți candidați. Candidații își vor pregăti CV-urile și participarea la interviu, iar angajatorii vor selecționa CV-urile, vor pune întrebări și vor alege unul sau mai mulți candidați. La sfârșit, discutați despre modul cum a decurs activitatea, stabilind ce ați făcut bine și unde trebuia să procedați în mod diferit. Realizați și susțineți o prezentare în fața colegilor de clasă privind activitatea voastră.

Teme opționale

- *Protecția bazelor de date*
- *Tehnici avansate de gestiune a structurii conceptuale a tabelor*

Protecția bazelor de date

Exploatarea actuală a datelor se face utilizând din ce în ce mai multe configurații cu un număr mare de utilizatori și cu volum mare de prelucrări. Majoritatea SGBD-urilor permit exploatarea bazelor de date distribuite pe mai multe mașini, în rețele eterogene de calculatoare, de către utilizatori foarte diverși.

Există două cerințe legate de protecția datelor cu care se lucrează:

1. asigurarea **confidențialității** informațiilor;
2. asigurarea **integrității datelor** prin protecția datelor în condițiile accesului multiutilizator și împotriva defectelor hardware sau a erorilor programatorilor.

Confidențialitatea datelor

Înseamnă protecția împotriva accesului neautorizat la bazele de date. Este rolul administratorului de sistem de a decide necesitatea unui sistem de securitate și de a menține integritatea acestuia. Înainte de a defini un plan de securitate, este necesar să fie detectate toate riscurile posibile și să fie analizate consecințele directe și indirecte ale riscurilor. Iată câteva mecanisme de asigurare a confidențialității datelor:

1. Parolele sau cuvintele de trecere (password) sunt organizate pe niveluri, pentru identificarea utilizatorilor și a drepturilor lor de acces. Pot fi acordate drepturi numai de citire sau/și de actualizare la întreaga bază de date sau la o parte a ei. Memorarea parolelor se face, de regulă, de către administrator într-un fișier care are drept câmpuri parola, utilizatorul și nivelul de acces. E bine ca numele dat acestui fișier să fie ales cu grijă. O bună idee ar fi criptarea parolelor.

2. Mecanismul de utilizare a fișierelor virtuale (imaginilor) adică a schemelor externe ale bazei de date – permite securitatea bazei în sensul restricționării la nivelul schemelor conceptuale a accesului la date al diferiților utilizatori.

3. Criptarea datelor este operația de codificare pe timpul stocării sau transportului, astfel încât descifrarea lor să fie posibilă numai de către posesorii cheii. În practică este frecvent utilizat sistemul DES (Data Encryption Standard), care are la bază un algoritm de criptare public, dar utilizează o cheie de criptare privată.

4. Metodele biometrice de identificare, autentificare și acceptare/refuz a intrării în sistem a unei persoane se bazează pe amprente digitale sau genetice, decodarea vocii sau studiul semnăturii etc. Sunt deja larg folosite în sistemele militare, dar pot pune unele probleme de etică.

5. Cartelele microprocesor conțin mecanisme de stocare și prelucrare a datelor sub controlul propriului circuit, la inițiativa fabricantului. Este o metodă sigură, căci nu permite nici accesarea, nici modificarea programelor și a datelor conținute pe cartelă.

Integritatea datelor

Un alt aspect legat de protecția datelor privește corectitudinea datelor introduse și manipulate și se concretizează prin:

- asigurarea integrității semantice a datelor;
- controlul accesului concurrent;
- salvarea și restaurarea bazei de date.

1. Asigurarea integrității semantice a datelor prin prevenirea introducerii unor date incorecte și a efectuării unor prelucrări greșite presupune includerea în programele de aplicație a unor secvențe de testare a datelor și respectarea restricțiilor de integritate a bazei.

2. Controlul accesului concurrent

Unul dintre principalele avantaje ale utilizatorilor de rețele de calculatoare este faptul că un program sau un fișier de date poate fi folosit de mai mulți utilizatori fără a fi nevoie ca fiecare dintre aceștia să aibă pe stația sa de lucru o copie a programului sau a fișierului respectiv. În acest mod se economisește spațiu de memorie și timp. Un alt avantaj major al folosirii bazelor de date în rețelele de calculatoare este accesul utilizatorului la *versiunea actualizată* a unei baze de date. Dar, în același timp, pot avea loc și situații conflictuale, când doi utilizatori doresc să modifice aceeași dată! Sistemul trebuie controlat astfel încât datele să nu fie compromise sau actualizate necorespunzător.

Sistemul FoxPro permite utilizarea bazelor de date în două moduri:

- modul **privilegiat sau exclusiv** (exclusive) – permite accesul unui singur utilizator la date;
- modul **comun sau partajat** (share) – permite accesul mai multor utilizatori.

În cazul utilizării comune a bazei de date de către mai mulți clienți (în mod concurrent) este necesară aplicarea mecanismelor de protecție a datelor față de accesul simultan. Regula generală este: *nu este permisă citirea unei înregistrări concomitent cu o scriere în aceeași înregistrare*. Deci înainte de citirea unor date dintr-un articol este necesară blocarea accesului la scriere a celorlalți utilizatori la înregistrarea respectivă.

A. Mecanisme de blocare

Blocarea se poate aplica la nivelul întregii baze de date, la nivelul unui fișier, al unei înregistrări sau chiar al unui câmp al articolului.

Tip de blocare	Nivel tabelă	Nivel articol	Nivel câmp
Implicită	Append	Append Memo	
	Delete All	Modi Memo	
	Recall All	Browse/Edit	
	Replace All	Replace Record	
	Update	Delete/Recall Record	
Explicită	Set Exclusive On	Flock()	Lock()
	Set Lock On		Rlock()

Exemple


Exemplul 1: Listarea unei tabele la imprimantă fără ca, între timp, altcineva să poată modifica fișierul.

```
use elevi
set lock on
list to printer
set lock off
use
```

Exemplul 2: Repetarea încercării de blocare a unei baze de date până când aceasta reușește.

```
set reprocess to 0
on error do eroare
use test
if flock()= .t.
sort on nume to man
unlock
use man
endif
```

procedure eroare
retry
return



B. Mecanismul tranzacțiilor

Un alt mod de protejare a datelor în rețea este organizarea operațiilor în tranzacții. O **tranzacție** constă dintr-o succesiune de comenzi elementare asupra unei tabele și a fișierelor index și memo care o însoțesc. Orice tranzacție este caracterizată printr-un punct de începere, o secvență de operații și un punct de sfârșit.

În rețea, eșuarea unei tranzacții poate avea loc din următoarele cauze: întreruperea curentului electric, eroare hardware la citire/scriere pe disc, eroare software, incident la cablu sau conector etc. Întreruperea executării unei tranzacții poate conduce la alterarea datelor.

Spunem că este asigurată **starea consistentă** a bazei de date atunci când baza reflectă rezultatele finale ale execuției unor tranzacții, nici o tranzacție nu este în curs de execuție și sunt respectate restricțiile de integritate semantică.

3. Salvarea și restaurarea bazei de date

Dacă prin funcționarea anormală a rețelei sau căderea SGBD-ului sau prin alte incidente hardware, tranzacțiile nu se încheie normal și există pericolul unei deteriorări a datelor, trebuie să apelăm la mecanismul de refacere a stării anterioare, consistente a bazei. Salvarea, din punctul de vedere al asigurării integrității datelor, este procesul de stocare a datelor în vederea restaurării contextului anterior în cazuri extreme și poate fi făcută prin:

- a) *copii de siguranță (backup) ale bazei de date* – sunt realizate automat, fie de către sistem, la anumite intervale de timp, fie de către administratorul bazei de date. De regulă, copia bazei de date este utilizată la căderea sau pierderea sau distrugerea suportului care păstrează baza, de aceea este bine să fie făcută pe alt suport (disc, bandă magnetică).

- b) *Jurnale de tranzacții* – reprezintă fișiere speciale, întreținute de monitorul SGBD, ce conțin o evidență riguroasă a modificărilor efectuate asupra bazei, prin contabilizarea următoarelor informații:
- *Identificatorul programului, utilizatorului și postului de lucru care face apelul;*
 - *Durata apelului;*
 - *Tipul modificării (adaugă, modifică, șterge) și articolele care au suferit aceste modificări;*
 - *Natura tranzacției sistem sau a aplicației;*
 - *Identificarea încălcărilor integrității bazei de date și semnalizarea erorilor către utilizator.*
- c) *Jurnale de imagini* – spre deosebire de jurnalele de tranzacții, nu conțin operațiile care au fost efectuate, ci efectul acestora. Pot fi:
- *jurnale cu imaginea datelor după modificare;*
 - *jurnale cu imaginea datelor înainte de tranzacție;*
 - *jurnale ce conțin ambele variante ale datelor înainte și după dezastru.*

Restaurarea bazei poate fi realizată automat de monitorul SGBD. Pe baza fișierelor jurnal, se poate face reorganizarea structurii bazei după accident, în sensul revenirii la starea anterioară accidentului. Restaurarea poate fi făcută și de către administrator, prin încărcarea copiei de siguranță celei mai recente, urmând ca apoi să fie reluate prelucrările.

Tehnici avansate de gestiune a structurii conceptuale a unei tabele

O aplicație la cheie trebuie să permită uneori construirea tabelor conform cerințelor particulare ale utilizatorilor, fără să le impună cunoașterea limbajului FoxPro. Printr-un dialog cu utilizatorul, aplicația informatică va trebui să obțină informațiile necesare la generarea structurii tabelii de date.

1. **Comanda CREATE FROM ARRAY** permite constituirea structurii unei tabele din informațiile unui tablou.

```
CREATE TABLE <fis.dbf> FROM ARRAY <tablou>
```

Comanda preia numele câmpurilor din prima coloană a masivului <tablou>, tipul din a doua coloană, lungimea fiecărui câmp din coloana a treia, iar numărul de poziții zecimale din coloana a patra. Este creată numai structura tabelii.

Operația inversă, de transfer al informațiilor de structură într-un tablou, este realizată de funcția Afields. Funcția AFIELDS (<tablou>) încarcă într-un tablou structura unei tabele active. În prima coloană sunt memorate numele câmpurilor, în coloana a doua tipul câmpurilor din structură, în coloana a treia lungimea și în coloana a patra numărul pozițiilor zecimale.

Exemplu

Să se creeze o nouă tabelă cu aceeași structură cu a fișierului ELEV1.DBF, dar având câmpurile sortate alfabetic, iar câmpul *nume* va fi înlocuit cu câmpul *prenume* (modificarea unui nume de câmp).

<code>use elev</code>	copiem în tabloul A structura fișierului elevi
<code>=afields(a)</code>	
<code>poz=ascan(a, 'NUME', 1)</code>	cautăm în tablou valoarea „NUME“
<code>if poz# 0</code>	Dacă se găsește
<code>a[poz]='PRENUME'</code>	înlocuim cu valoarea „PRENUME“
<code>endif</code>	
<code>=ASORT(a)</code>	sortăm tabloul alfabetic după prima coloană
<code>create table b from array a</code>	copierea tabloului ca structură

2. Comanda **CREATE FROM** transformă o tabelă specială (cu structură extinsă) într-o structură obișnuită în tabela `<fis.dbf>`

```
CREATE <fis.dbf> FROM <fis.dbf extins>
```

Sub numele de structură extinsă este folosită o tabelă specială având câmpurile: **FIELD_NAME**, **FIELD_TYPE**, **FIELD_LEN**, **FIELD_DEC** și drept articole tocmai câmpurile tabelii.

3. Comanda **COPY STRUCTURE EXTENDED**, cu formatul dat în continuare, folosește tabela activă și creează structura extinsă pentru fișierul `<fis.dbf extins>`

```
COPY TO <fis.dbf extins> STRUCTURE EXTENDED
```

Baza de date cu structură extinsă nu este ștersă automat prin această manevră.

Exemple

Exemplul 1. Fie baza de date ELEV1 (nume, pren, cls, media, absn, absm, dn). Vrem să realizăm o altă tabelă care să aibă numai primele trei câmpuri și ultimul (nume, pren, cls, dn) fără să folosim denumirile acestor câmpuri¹.

Varianta 1:

```
use elevi                                recall record reccount()
copy to man structure extended           pack
use man                                  create nume from man
go 4                                      use man
dele rest                                 append from elevi
```

&& observați tabela realizată prin comanda **COPY EXTENDED**:

#	FIELD_NAME	FIELD_TYPE	FIELD_LEN	FIELD_DEC	FIELD_IDX
1	NUME	C	10	0	Y
2	PREN	C	6	0	N
3	CLS	C	3	0	Y
4	ABSN	N	2	0	N
5	ADMIS	L	1	0	N
6	DN	D	8	0	N

¹ Comanda **Copy to** cere precizarea câmpurilor care vor fi copiate.

Varianta 2

```
use elevi
for i=1 to 3
x=field(i)
set fields to &x
  && lista de campuri
endfor
copy to man
  &&se va copia structura
use man
append from elevi
```

Varianta 3

```
use elevi
=afield(a)&& copiem structura
n=alen(a,1)
for i=1 to n-4
=adel(a,4,1)&&stergem coloanele
endfor
dimension a^4,mp
create table man from array a
use man
```

Exemplul 2. Dorim o aplicație pentru concursuri școlare. Procedurile sunt aceleași: înscrierea concurenților, introducerea notelor la fiecare probă, calcularea mediei și determinarea reușităților. Ce diferă însă de la un concurs la altul? Numărul de probe, denumirea fiecărei probe, poate sistemul de notare, modalitatea de a determina reușitățile.

Deci ne propunem ca prima procedură din acest program să fie cea de **construire a tabelii particulare, specifice, unui anumit concurs** (probele de examen sunt notate de la 1 la 10). Urmăriți variantele de rezolvare:

procedure creare_1

```
use manevra
set fields to nume, mg
input "NR. DE PROBE:" to n
for i=1 to n
var='ob'+ltrim(str(i, 2))
set fields to &var
endfor
copy stru to concurs
use concurs
return
```

procedure creare_2

```
use manevra
copy stru exte to man
use man
delete all for field_name #'NUME' and. ;
  field_name#'MG'
input "numar probe? " to n
for i=1 to n
var='OB'+ltrim(str(i))
recall for field_name=var
endfor
pack
create concurs from man
use concurs
return
```

procprocedure creare_3

```
create table X(nume C(20),
  mg N(5,2))
copy stru exte to man
use man
input "numar probe? " to n
for i=1 to n
var='ob'+ltrim(str(i))
append blank
repl field_name with var
repl field_type with 'n'
repl field_len with 5,field_dec
  with 2
endfor
create concurs from man
use concurs
erase X.dbf
erase man.dbf
return
```

procedure creare_4

```
create table X(nume C(10), mg
N(5,2))
copy stru extended to man
use man
input "numar probe? " to n
for i=1 to n
append blank
accept "obiect?" to field_name
repl field_type with;
  'N', field_len with 2
endfor
create concurs from man
use concurs
delete file x.dbf
delete file man.dbf
return
```



sarcini de laborator

1. La proiectarea structurii unei tabele necesare înregistrării persoanelor ce lucrează la firma AIRFRANCE S.A., câmpul adresa a fost dimensionat la 200 caractere. *Modificați structura fișierului* astfel încât dimensiunea acestui câmp să fie cât mai mică, pentru a păstra informațiile inițiale fără trunchiere și fără spații inutile.

2. Scrieți un program care să creeze fișierul FIS.DBF (nume C(8), ext C(3)) cu toate tabelele din directorul curent și să afișeze toate fișierele care au *structură identică*.

3. La cantina școlii sunt înregistrate intrările și consumurile de alimente din fiecare zi în fișierul ALIMENTE (data, cod-aliment, cod-mișcare, cantitate), unde cod-mișcare este fie literele „e” =consum, fie „i” = intrări alimente).

Știm că în fișierul STOCURI (nume-aliment, cod, preț, stoc) codificarea s-a făcut odată cu sosirea unui nou aliment pe baza numelui și a prețului acestuia. Să se afișeze situația următoare:

SITUAȚIA CONSUMULUI DE ALIMENTE IN LUNA...							
cod/nume-aliment	pret	z1	z2	...	31	total-cant	total-val
total/zile	-	Xx	xx	...	xx	xxxxxxx	xxxxxxx

Alimentele vor fi trecute în ordinea alfabetică, pe prețuri. Coloanele Z1-Z31 vor conține totalul cantităților de alimente consumate în zilele lunii solicitate. Pentru fiecare aliment se calculează totalul cantitativ (coloana **total cant**) și valoric în lună (coloana **total-val**). Pentru fiecare zi se calculează totalul valoric în ultima linie, notată **total/zile**.

4.² La un oficiu poștal se ține evidența revistelor la care se poate face abonament în fișierul REVISTE(cod_rev N(3), nume_rev C(10), pret N(6)). În momentul sosirii unui client care dorește abonament, se alege revista și se completează perioada, numărul de bucăți solicitate, datele personale ale abonatului în tabela ABONAM(data D, nr_chit N(5), cod_rev N(3), nume_pers C(10), Adr M, perioada C(5), nr_buc N(3)).

Se cer:

1. O procedură de introducere a datelor la abonarea unui client;
2. O procedură care să creeze structura unui fișier numit CENTRALIZ:

data	nr_chit	nume_pers	adr	nume_rev	L01	L12
------	---------	-----------	-----	----------	-----	------	-----

3. O procedură de încărcare cu date a fișierului CENTRALIZ din fișierele ABON și REV;
4. O procedură pentru calcularea valorii totale lunare a tuturor abonamentelor pentru revista X;
5. O procedură pentru afișarea numărului total de exemplare cerute prin abonament pentru fiecare revistă;
6. O procedură pentru aflarea cheltuielii făcute de o persoană X.

² Vedeti o variantă de rezolvare la sfârșitul cărții.

Dezvoltarea unei aplicații informatice

- *Aspecte teoretice*
- *Proiect final, sugestii de rezolvare*

Aspecte teoretice

Ingenieria software sau ingineria produselor program (*software engineering*) este un domeniu care încearcă o abordare sistemică a proiectării, operării, întreținerii și dezvoltării produselor informatice. Numele vine din alăturarea cuvintelor *software* (care reprezintă atât codul sursă al unui program, cât și documentația corespunzătoare) și *engineering* (care se referă la aplicarea unei abordări sistematice, științifice a problemei, abordare care are drept scop rezolvarea corectă și cu minim de efort a problemei).

Principii în dezvoltarea software

- procesul de realizare a unui produs informatic de o anumită complexitate poate și trebuie să fie structurat într-o serie de etape/subetape/activități bine determinate;
- în realizarea produsului informatic sunt antrenate diferite persoane, nu numai informaticieni; de aceea, este necesară o bună organizare și specializare, o repartizare precisă a sarcinilor; se recomandă folosirea tehnicilor speciale, adecvate dimensiunii proiectului și etapei respective;
- trebuie să existe o documentare bună, într-o formă standardizată, a fiecărei faze parcurse;
- procesul trebuie să fie controlat tot timpul, iar abaterile constatate, împreună cu propunerile de remediere, trebuie semnalate celor care-l conduc și răspund de buna lui desfășurare;
- pentru a avea siguranța că producem ceva util, trebuie să se țină seama de modificările care apar în timpul desfășurării procesului, în primul rând cele cu caracter organizatoric și tehnologic, precum și de posibilele dezvoltări viitoare;
- în toate fazele de dezvoltare a produsului trebuie antrenat beneficiarul lucrării, cel care va plăti de fapt efortul. Acesta decide anumite variante de continuare a lucrărilor, se obișnuiește cu sistemul informatic, își poate clarifica anumite cerințe la timp și nu după ce produsul este gata. Acest principiu este foarte important pentru acceptarea lucrării și conduce la o scurtare a perioadei de implementare și a costurilor;
- criteriul principal care trebuie să stea la baza realizării produsului este cel economic, trebuie estimate corect costurile și beneficiile;

- Produsele informatice nu-și dovedesc viabilitatea decât în măsura în care reunesc cerințele și interesele utilizatorilor. Ei dictează ce trebuie informatizat și când anume. Se conturează astfel anumite condiții de acceptare a produselor soft: **funcționalitate** (să rezolve problemele organizației), **rentabilitate** (legat de impactul economic provocat de introducerea sistemului), **adaptabilitate** (capacitatea sistemului de evolua conform contextului) etc.

Ce doresc utilizatorii?

- răspunsuri exacte și clare la probleme foarte diferite;
- comunicare directă cu calculatorul în limbaje accesibile, de nivel conversațional;
- independență a aplicațiilor față de sistemul de calcul, astfel încât schimbarea calculatorului să nu impună schimbarea softului;
- produse compatibile cu cele anterioare și standarde (mai ales pentru comunicații);
- controlul asupra datelor proprii;
- un mod variat de introducere a datelor și de obținere a rezultatelor (voce, imagine, text);
- un preț mic și o durată de realizare mică.

Ingineria produselor informatice conține metode, tehnologii (instrumente) și proceduri pentru conducerea și controlul producției de software. Unii autori numesc acest domeniu de activitate *metodologia* sau *tehnologia realizării produselor program*.

Etapele principale de realizare a produselor informatice

Efortul de realizare a unui software este declanșat de cererea unui client. Ca urmare a descoperirii unor **probleme**, aspecte negative în instituția unde lucrează sau pe care o conduce, clientul dorește realizarea unui produs informatic care să rezolve problemele. O altă situație care poate declanșa cererea de soft poate fi dorința folosirii tehnicii noi de calcul sau a unor metode noi de conducere (programare dinamică, simulare etc.).

Etapa 1. Formularea problemei

Primul pas este luarea contactului cu clientul pentru clarificarea domeniului în care va fi implementat produsul informatic, clarificarea problemei puse, a motivelor, a costurilor estimative, a unor variante de rezolvare și inventarierea efectelor. Este un moment decisiv pentru că, uneori, clientul își dă seama că nu poate suporta financiar efortul de dezvoltare a software-ului și renunță.

Atenție! O problemă nu este niciodată complet definită, există întotdeauna cel puțin câteva elemente **subînțelese**, datorită cunoștințelor din domeniul investigat pe care le are cel ce pune problema, cunoștințe care nu sunt cunoscute de programator. Prin urmare, un prim efort trebuie făcut spre enunțarea clară, corectă și completă a problemei de rezolvat.

Etapa 2. Analiza diagnostic a situației existente

Este etapa cea mai importantă pentru că permite stabilirea exactă a cerințelor, obiectivelor și limitelor produsului informatic. Acest lucru se face printr-un studiu complet al activităților desfășurate în domeniul problemei, dar și al activităților conexe. Prin interviuri, chestionare și observări directe, prin cercetarea tuturor documentelor, actelor etc. sunt clarificate cerințele clientului și sunt stabilite posibile variante de rezolvare.

Prima etapă de analiză este **culegerea informațiilor** despre sistemul informatic existent; este cea mai lungă și foarte importantă. Scopul este cunoașterea detaliată a:

- a) informațiilor, surselor și consumatorilor de informații;
- b) principalelor activități legate de date: prelucrări, reguli de calcul, validări, puncte de control, modalități de arhivare;
- c) principalelor acte normative, legi, decrete etc. care restricționează sistemul nou (de exemplu, documentele tipizate au un conținut și un traseu deja fixat, care nu poate fi modificat);
- d) sistemului de codificare existent;
- e) mijloacelor tehnice existente;
- f) personalului implicat în fluxurile informaționale: număr, calificare etc.

Culegerea tuturor informațiilor despre activitățile desfășurate, modalitățile de conducere a acestor activități și despre sistemul informațional existent se finalizează printr-o **analiză critică** a situației existente: depistarea aspectelor negative și identificarea **cerințelor și obiectivelor** noului sistem informatic, precum și a limitelor și restricțiilor de realizare.

În evaluarea sistemului, soluțiile spontane reprezintă o metodă ideală de **concepere a scenariilor de tipul „ce-ar fi dacă?”**. O decizie tipică oscilează între varianta de îmbunătățire a sistemului existent prin schimbarea fluxurilor de informații, reducerea documentelor, schimbarea sarcinilor și varianta de informatizare. Avantajul celei de a doua variantă în ceea ce privește flexibilitatea, acuratețea, promptitudinea cu care se raportează orice fel de informații trebuie pus în balanță cu efortul (nu numai financiar) pentru realizarea acesteia.

Studiul și analiza sistemului existent se fac prin intermediul mai multor **tehnici**, dintre care amintim: tehnica interviului, tehnica chestionarelor, studiul documentelor, observația directă, tabele de decizie. În culegerea și analiza datelor despre sistemul existent, dar și în următoarele etape, sunt folosite tehnici de reprezentare grafică a fluxurilor de date, evenimente, structuri etc. Schemele și diagramele au avantajul că sunt simple, redau logica sau cronologia prelucrării fără ambiguități și pot fi ușor interpretate. O imagine este mai sugestivă decât o mie de cuvinte.

a. Tehnica interviului

Scopul tehnicii interviului este obținerea într-un timp redus a informațiilor privind orientările practice, metodele și strategiile existente sau preconizate în rezolvarea problemelor analizate. Interviu asigură:

- obținerea informațiilor cât mai recente și cât mai complete;
- compararea și verificarea informațiilor obținute prin alte metode sau tehnici;
- culegerea unor date semnificative la locul producerii lor.

Interviul poate fi folosit cu succes în toate etapele de concepere, realizare și funcționare a sistemelor informatice, dar trebuie subliniată necesitatea îndeplinirii a două condiții pentru ca rezultatele obținute în urma unui interviu să fie cu adevărat utile:

- a. menținerea în timpul interviului a unei atmosfere de încredere și acceptare, cu mare atenție de a nu trezi respondentului teama că e forțat să dezvăluie informații (strict) confidentiale;
- b. purtarea discuțiilor în așa fel încât interlocutorul să înțeleagă și să simtă că ideile lui vor afecta lucrările care urmează să fie realizate, deci că opiniile sale vor fi respectate.

Procedura de aplicare a tehnicii interviului cuprinde mai multe etape și presupune respectarea mai multor condiții. Astfel:

1. *pregătirea interviului* – se realizează înaintea întâlnirii, când face o listă cu principalele întrebări, punctele critice ale activității asupra căreia dorim informații, programul de lucru etc.;
2. *interviul* – este preferată o atitudine neutră, nu foarte familiară, nici distantă; se iau notițe sau se înregistrează convorbirea dacă interlocutorul acceptă, iar la sfârșit sunt recapitulate punctele principale ale discuției, pentru a verifica dacă au fost înțelese cerințele clientului. Nu uitați să mulțumiți și să cereți acceptul unei eventuale revederi dacă mai apar aspecte de clarificat;
3. *după interviu* – transcrieți notițele, centralizând datele culese. Identificați punctele rămase nelămurite.

Concluziile interviului reflectă starea elementelor sau a proceselor analizate și posibilități de remediere a unor deficiențe existente.

b. Tehnica chestionarului

Tehnica chestionarului este recomandată când se dorește informarea asupra părerii generale despre un anumit aspect, culegerea informațiilor de la un număr mare de subiecți în același timp, despre aceeași problemă.

Tehnica chestionarului este folosită când cantitatea de informații care trebuie culeasă este redusă sau trebuie verificate anumite informații obținute prin interviuri sau observări directe. Fiind anonime, chestionarele încurajează subiecții în completarea răspunsurilor corect și cu simț critic (autocritic mai puțin!!) dar se pierde legătura afectivă cu persoana care răspunde și, evident, se evită răspunsurile la întrebările care deranjează.

c. Tehnica observării directe

Observarea directă a activităților, în special a activităților legate de sistemul informațional, asigură cunoașterea nemijlocită a situației existente. Se studiază sarcinile care formează conținutul unei activități. De exemplu: dotarea cu tehnică de calcul și folosirea acesteia, circulația informațiilor, modalități de stocare, timp de lucru etc.

Dacă obiectivele propuse prin observare directă sunt clar precizate, atunci aplicarea acestei tehnici duce la concluzii reale, care nu pot fi obținute prin alte metode sau tehnici. De exemplu, informațiile cu caracter tehnic pot fi confruntate cu rezultatele observării locurilor de muncă și a modului de utilizare a capacităților de producție.

Tehnica observării directe se bazează pe instrumente specifice de lucru: sondaje, cronometrări. Prin aplicarea lor se relevă stări de fapt, deficiențele existente și posibilități de ameliorare.

Etapa 3. Proiectarea

Urmărește conturarea unui model al viitoarei aplicații, atât al datelor cât și al funcționalităților. Soluția informatică aprobată în etapa precedentă este structurată pe componente relativ independente, sunt determinate resursele necesare și termenele de realizare. Tot acum sunt estimate eficiența și impactul asupra oamenilor din organizația în care va fi implementat produsul. Etapa conține:

- a) Proiectarea bazei de date, a tabelelor, a relațiilor dintre acestea, a regulilor de validare, a condițiilor de integritate referențială etc.;
- b) Proiectarea interfețelor de intrare: documentele noi care intervin în sistemul informațional, fluxurile de informații de la sursele de date către calculator, precum și formularele de introducere a datelor în calculator;
- c) Proiectarea interfețelor de ieșire: documentele de raportare, graficele, precum și fluxurile acestor documente către utilizatori;
- d) Proiectarea prelucrărilor automate: proceduri sau interogări directe, modalitățile de apel;
- e) Proiectarea programului monitor și a meniului principal al aplicației.

Etapa 5. Construirea și testarea aplicației

Etapa de construire poate fi unită cu etapa de proiectare pentru lucrările de mai mică amploare, mai ales prin folosirea unui mediu vizual de dezvoltare. Însă testarea produsului rămâne deosebit de necesară, folosind date fictive, în vederea depistării și corectării erorilor.

Etapa 6. Implementarea și experimentarea cu date reale a aplicației

Este realizată prin metode specifice dimensiunii și complexității aplicației informatice, în vederea validării prin practică. Pentru fiecare aplicație, este experimentată funcționarea în condiții reale în paralel cu vechea procedură de lucru pentru o anumită perioadă de timp. Dacă rezultatele sunt cele așteptate, atunci se declară sistemul operabil și echipa de specialiști care l-a realizat lasă exploatarea aplicației pe seama beneficiarilor.

Etapa 7. Exploatarea curentă, întreținerea și dezvoltarea

Activitatea de exploatare curentă a sistemului informatic este evaluată permanent, pentru a depista corecțiile necesare. Corecțiile pot fi de mică sau mare amploare și formează activitatea de întreținere sau dezvoltare a sistemului informatic.

Dacă însă apar anumite schimbări tehnologice, restructurarea unității sau noi metode de conducere, atunci **se impun o nouă analiză și strategie.**

Documentarea aplicațiilor informatice

Trecerea de la artizanat la industrie în activitatea software a însemnat cerințe noi și pe linia documentării aplicațiilor. Programele generalizabile, folosite de un număr tot mai mare de utilizatori, trebuie să fie însoțite de o documentație clară și completă, care să conțină informații în legătură cu funcția programului, structura generală și de detaliu a acestuia, datele de intrare și de ieșire, specificațiile de realizare, implementare și exploatare. Se apreciază că aproape 10% din fondul de timp afectat activității de programare este utilizat pentru activitatea de documentare¹.

În funcție de destinația lor, distingem trei tipuri de documentație:

A. Documentația destinată **utilizatorului**, care include:

- *documentația de prezentare a aplicației* – conține informații generale, de natură tehnică și economică, asupra produsului program în întregul său; este destinată potențialilor utilizatori; are un pronunțat caracter comercial; insistă asupra posibilităților oferite de produsul program, eventualele performanțe care recomandă produsul dintre mai multe programe similare existente pe piață;
- *documentația de utilizare* – conține informații privind utilizarea curentă a produsului program: aria de probleme acoperită (limite, restricții), descrierea intrărilor și ieșirilor, procedurile de codificare și validare, procedurile de interpretare a ieșirilor, organizarea datelor, descrierea funcțională a procedurilor, metodele folosite, estimarea performanțelor, exemple.

B. Documentația destinată **echipei de programatori**. Constituind principalul mijloc de comunicare între diversele categorii de specialiști din echipa de realizare, documentațiile formulează problemele care au fost rezolvate în etapa/subetapa/faza curentă și problemele ce urmează a fi rezolvate în etapele următoare.

C. Documentația destinată **echipei de întreținere și dezvoltare** a produsului program, care cuprinde rezultatele activității de elaborare și testare a procedurilor. Se numește *specificație de realizare*.

D. Documentația destinată **personalului unității implicat în exploatare** (operatori, verificali, administratori). Se numește *manualul utilizatorului*.

Specificațiile sunt – pe plan mondial – din ce în ce mai apropiate de o formă standard și, de regulă, când se pronunță acest termen, un specialist știe despre ce este vorba. Uneori, la proiectele mici, cu scopul declarat de a întocmi un singur tip de specificații – cele de realizare – etapele de proiectare logică și tehnică se pot contopi într-una singură. Dar, atenție, sunt necesare persoane cu experiență, care să nu fie tentate să sară peste unele activități obligatorii.

Documentația trebuie să fie concepută **modular**, în așa fel încât fiecare categorie de personal implicată în funcționarea sistemului informatic să o poată folosi în mod independent; să fie clară, sugestivă, editată într-o formă grafică adecvată, cu exemple de utilizare, ținând seama de pregătirea de bază a fiecărei categorii de personal, să fie întocmită în colaborare cu personalul beneficiar, **receptată și însușită** de acesta înaintea etapei de implementare.

¹ După „Ingineria programării“ vol. 2, pag. 53

Proiectul final

Trebuie să realizați la nivelul fiecărei echipe de 2-3 elevi câte un proiect final, care să demonstreze că aveți atât cunoștințele necesare dezvoltării unui proiect cu baze de date, cât și deprinderile de a lucra în echipă și de a vă prezenta lucrarea.

Ce se cere?

1. să alegeți o problemă din lumea reală;
2. să modelați baza de date necesară rezolvării problemei;
3. să dovediți cunoștințele necesare proiectării bazei de date, a formularelor, interogărilor, formularelor și rapoartelor în FoxPro;
4. să elaborați documentația proiectului (Word+PowerPoint);
5. să prezentați lucrarea în fața colegilor și a profesorului vostru.

1. Activitatea de formulare a problemei

Vă sugerăm ca, studiind exemplele din lumea reală, din manual sau de pe Internet, să construiți un scenariu, o problemă care necesită modelarea unei baze de date. De exemplu, sunteți patronul unei cafenele și doriți să știți care este consumul de cafea pe lună, care lună este mai aglomerată, dacă trebuie să mai angajați personal sau nu, dacă aveți clienți nemulțumiți, dacă ați avut sau nu profit!

2. Analiza situației existente

Întocmiți fișe de analiză, identificând activitățile care pot fi informatizate: documentele specifice purtătoare de informații, fluxurile informaționale, prelucrările specifice. Faceți o critică a situației existente. Imaginați mai multe variante de rezolvare a problemei și analizați efectele fiecăreia. Alegeți o variantă. Desenați schema conceptuală a bazei de date. NU folosiți mai mult de 5 entități. NU sunt necesare mai mult de 5 interogări sau rapoarte!

4. Proiectarea aplicației

Identificați fluxurile de prelucrări, intrările și ieșirile, legăturile dintre ele. Stabiliți entitățile cu proprietățile (atributele, tipul și lungimea) lor și cerințele de validare. Determinați legăturile dintre entități; normalizați. Proiectați formularele pentru culegerea datelor, pentru vizualizare și editare. Proiectați forma rapoartelor utilizator: situații centralizatoare sau grafice. Proiectați panoul de bord – cum va fi prezentată aplicația, cum vor fi selectate operațiile specifice acesteia: meniuri, butoane, taste funcționale. Proiectați schema de sistem a aplicației.

5. Construirea și testarea aplicației

Este momentul să creați baza de date cu tabelele ei și să impuneți restricțiile de integritate. Populați baza de date cu date de test. Construiți formularele de culegere și vizualizare a datelor. Construiți interogările. Atenție la informațiile cerute, dacă sunt corecte și complete, dacă forma lor este cea optimă (alegeți tabele sau diagrame). Construiți machetele rapoartelor. Uniți componentele aplicației într-un tablou de bord sau meniu.

6. Implementarea aplicației

Pregătiți datele de test și realizați testarea componentelor pe rând, apoi integrați-le în proiectul final. Realizați un grafic de implementare.

7. Documentarea

Fiecare echipă va realiza o documentație de prezentare a rezolvării problemei în fața „clientului” (în PowerPoint) și o documentație de utilizare, cu detalii despre fiecare componentă în parte (în Word).

8. Prezentarea proiectului

Trebuie să demonstrați că știți să lucrați în echipă, că fiecare v-ați asumat un anumit rol și l-ați realizat foarte bine. Începeți prin a prezenta membrii grupei, apoi „problema”. Puteți preciza restricțiile și variantele de soluții propuse (dacă sunt). Accentul ar trebui pus pe explicarea clară a modelului conceptual al bazei de date (SCHEMA).

Aplicația trebuie să fie modularizată, deci fiecare modul (fereastră) ar trebui să fie explicat „clientului” prin funcționalitatea sa.

Prezentați documentația scrisă. Puteți sugera și o posibilă dezvoltare a proiectului. Asigurați-vă că vă încadrați în timp. La sfârșitul prezentării, răspundeți la întrebările celorlalte grupe.

Vă reamintim cele 10 reguli pentru susținerea cu succes a unei prezentări:

1. relaxați-vă și respirați adânc;
2. stați drept, adoptați o atitudine de încredere în ceea ce veți spune;
3. concentrați-vă asupra mesajului pe care doriți să-l transmiteți, nu vă lăsați perturbat de alte lucruri;
4. folosiți un limbaj accesibil, corect;
5. păstrați contactul vizual cu câteva fețe prietenoase și captați-le atenția asupra subiectului pe care îl dezvoltați;
6. nu vă lăudați și nici nu vă cereți scuze;
7. încadrați-vă în timp;
8. nu vă jucați cu indicatorul pe planșă sau pe diapozitivul pe care îl prezentați;
9. nu puneți mâinile în buzunare;
10. îmbrăcați-vă corespunzător.

Notă. Este foarte important să discutați după prezentarea proiectului final următoarele aspecte: Ce parte din ciclul de viață a dezvoltării unei aplicații informatice vi s-a părut mai grea? Ce etapă a durat mai mult? Preferați să analizați cerințele problemei, sau să construiți tabelele sau să scrieți documentația? Ce v-a plăcut mai mult? Care funcție?

Exemple de probleme

1. Firma Cori_Flame este specializată în producerea și comercializarea produselor cosmetice și își organizează vânzarea către populație prin intermediul distribuitorilor autorizați.

Fiecare persoană care dorește să fie distribuitor trebuie să fie recomandat de un distribuitor deja autorizat, care va fi numit sponsor. La înscriere, persoana viitor distribuitor semnează un *contract* de fidelitate, comunicând codul sponsorului și primește un cod sau număr de distribuitor, care va fi folosit în toate tranzacțiile cu firma. Codul este astfel realizat încât să permită legătura cu sponsorul. Un distribuitor este activ dacă are cel puțin o comandă în ultimele 6 luni. Distribuitorul prospectează piața și face o *comandă* către firmă, angajându-se să comercializeze anumite produse în anumite cantități. Pe comandă va fi notat codul distribuitorului și, pentru fiecare produs, codul și cantitatea cerută. Marfa cerută nu poate fi returnată decât dacă are defecte. Operațiile de vânzare a mărfii se realizează de către distribuitor, care primește marfa cerută prin comanda sa odată cu documentul numit *factură*. În factură sunt identificate produsele, cantitățile trimise, precum și prețul de distribuție (separat se trece TVA). Factura are un număr și o dată. Factura trebuie achitată în termen de 30 zile, altfel se percep penalizări. Produsele comercializate de firma Cori_Flame au un preț de distribuție (pe care îl plătesc distribuitorii la achitarea facturii), un preț de vânzare (la care poate fi comercializat către populație) și un punctaj pentru stabilirea unei bonificații de stimulare a vânzării. Pe parcursul unei luni sunt totalizate facturile achitate de un distribuitor și punctajul de vânzare. Nu este permisă înregistrarea unei noi comenzi de către un distribuitor care nu și-a achitat factura anterioară. Comenzile servesc și la orientarea producției. De obicei, mărfurile solicitate se află în stocul depozitului firmei, dar uneori cererea poate depăși stocul existent, deci vor fi livrate și facturate mărfurile (în ordinea înregistrării cererilor), în funcție de cantitățile existente.

Firma dorește:

- evidența distribuitorilor (câți sunt înscriși, activi, cu credite restante etc.);
- operații de înscriere a noi distribuitori, retragere la cerere, scoatere din evidență pentru distribuitorii inactivi în ultimul an;
- gestiunea produselor comercializate de firmă, prin înregistrarea intrărilor și a ieșirilor, modificări de prețuri sau punctaje;
- gestiunea cererilor (înscriere cerere, facturare cu scoaterea cererii din evidență);
- gestiunea facturilor (trimiteri, achitări).

2. Sunteți angajat al unei companii specializate de recrutare și selecție a personalului. Orice firmă trimite oferta de posturi disponibile, cu avantajele și cerințele fiecăruia. Candidații au acces la oferte și își trimit CV-ul.

Dumneavoastră aveți sarcina de a asista firmele în decizia de angajare, ordonând CV-urile primite în funcție de cerințele postului.

Cum veți face?

1. Vă documentați pe Internet și vedeți ce posturi în domeniul IT sunt oferite în țară.
2. Pentru fiecare post, faceți o listă cu avantajele oferite: salariu, transport, concedii, condiții de lucru.
3. Pentru fiecare post, faceți o listă cu cerințele postului: studii, experiență, calificări suplimentare.
4. Memorați într-o bază de date persoanele care se înscriu la concurs, înregistrând CV-ul fiecăreia, poza, telefonul.
5. Notați observațiile după interviu.
6. Afișați CV-urile în funcție de studii.
7. Ordoanați CV-urile după experiență.

Sarcini suplimentare

1. Completați-vă CV-ul personal!
2. Faceți o listă cu întrebările interviului pentru ocuparea unui post în domeniul IT.

STUDIUL DE CAZ I

Analiza diagnostic, proiectarea bazei de date și a formularelor de introducere

- Folosirea formularelor și a schemelor pentru clarificarea problemei
- Imaginarea unor variante de rezolvare și analiza consecințelor fiecăreia

Problema

La casa de schimb valutar S.C. PATRICIA S.R.L. este necesar un program care ajută agentul de schimb la desfășurarea activităților sale. Astfel, agentul primește dimineața cursul valutar și sumele disponibile din fiecare monedă. Pe parcursul zilei, desfășoară operații de cumpărări și vânzări de valută, iar seara afișează situații centralizatoare, precum:

- lista tranzacțiilor din ziua respectivă;
- situația financiară: stoc inițial în lei la începutul zilei, total lei intrați prin operații de vânzare valută, total lei plătiți din casă prin operații de cumpărare valută, stocul final;
- lista operațiilor de tip vânzare valută, pe tipuri de monede, cronologic.

Să se realizeze o aplicație care să permită atât gestiunea diferitelor monede, cât și a tranzacțiilor efectuate la o **casă de schimb valutar**.

Analiza diagnostic

F1. Cine și ce face?

Prezentăm principalele sarcini legate de problemă, pe posturi de lucru, urmărind care pot fi reduse, eliminate, schimbate sau automatizate.

<i>Cine?</i>	<i>Ce face?</i>	<i>Perioada</i>	<i>Poate fi automatizat?</i>
Șeful	Hotărăște CE valute se tranzacționează și prețurile de vânzare-cumpărare ale acestora.	<i>Zilnic</i>	<i>Nu, deocamdată.</i>
Operator	Afișează cursul valutar.	<i>Zilnic</i>	<i>Da, dacă avem informațiile primite de la șef.</i>
Operator	Decide dacă un client poate fi refuzat; înregistrează operațiile de vânzare-cumpărare de valută	<i>Zilnic</i>	<i>Da, știind că o tranzacție se referă numai la valute care pot fi comercializate. Cantitatea cerută va fi</i>

în documente primare – numite buletin de schimb valutar.

comparată cu stocul de valută existent; oferta de valută va fi comparată cu suma de lei disponibilă.

Operator Afișează situații centralizate pe tipuri de tranzacții, pe valute etc. **Zilnic** *Da, dacă sunt memorate toate datele despre o tranzacție!*

Operator Îndosariază copiile după buletinele de schimb, cursul valutar și situațiile centralizate în dosarul zilei. **Zilnic** *Da, vom salva activitatea unei zile în istoric.*

F2. Care sunt documentele?

Ca purtătoare de informații, trebuie să inventariem toate documentele care sunt primite sau sunt completate în sistemul informatic analizat.

Cod	Nume document	Volum, tip	Observații
Bc	Buletin de schimb	200 exemplare/zi; de ieșire.	Elaborat de operator; certifică desfășurarea tranzacției; un exemplar este dat clientului și unul este păstrat la sediul firmei.
Listt	Situația tranzacțiilor	1/zi de ieșire; se introduce în dosarul zilei.	Elaborat de operator; folosește la evidența centralizată a tranzacțiilor; este trimis șefului.
listv	Cursul valutar	2/zi; de ieșire; unul este afișat, iar unul se reține la dosarul zilei.	Elaborat de operator; folosește la situația centralizată și la informarea clienților; este trimis șefului.

F3. Cum arată documentele?

Exchange Office „Patricia”
Iași, str. Vasile Conta, nr.7
CURSUL VALUTAR
Azi, 1.01.2003

Nrcrt	Valuta	Cumparare	vanzare
1	Dolar	30000	3200
2	Euro	28000	29000

Exchange Office „Patricia”
Iași, str. Vasile Conta, nr.7
Buletin de schimb
Nr 1234/din 1.01.2003, ora 14:30

Client POPESCU MIRCEA cu CI 123456 a cumpărat cantitatea de 100 dolari la cursul de 40000 LEI/USD
Suma achitată clientului = 100 USD
Suma încasată de la client = 4000000 LEI
Semnătura,

VALUTE			
01/17/03			
Cod Valuta:	1	Prez Camp:	Prez Vanz:
Nume Valuta:	dole	32,000	35,000
Cod Client	Cod Valuta	Tip Tranzactie	Cantitate
1	Tucano gh.	I C	10
23	Fluor mah.	I C	20
24	Pasaca d	I V	40
25	Jana OHT	I V	50

Proiectarea bazei de date

Pentru studiul de caz observăm că există două categorii de informații pe care să le gestionăm: despre valute și despre tranzacțiile cu aceste valute. Legătura între cele două entități este de tipul 1-n: o tranzacție se referă la o singură valută; pentru o valută pot fi înregistrate mai multe tranzacții.

Restricții și limite

1. Presupunem că vom solicita doar numele clientului – nu și buletinul, adresa etc.
2. Păstrăm tranzacțiile în ordinea sosirii lor – nu înregistrăm ora!

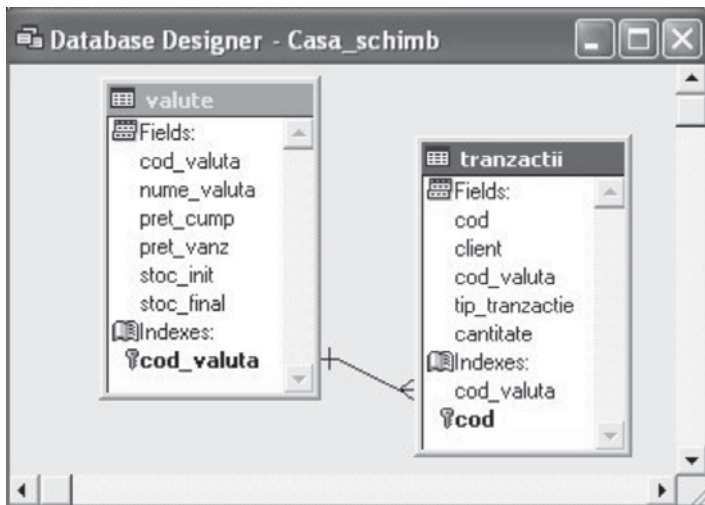


Figura S1-1: Tabelele valute și tranzacții

Proiectarea formularului pentru introducerea unei tranzacții

Client	valuta	Tip_tranzactie	Cantitate
<input type="text" value="btClient"/>	<input type="text" value="Combo1"/>	<input type="text" value="btTi"/>	<input type="text" value="btCantitate"/>
<input type="button" value="calculare"/>	<input type="text" value="Pret_cump"/>	<input type="text" value="Pret_vanz"/>	<input type="text" value="stocul de valuta"/>
	<input type="text" value="btPret_cump"/>	<input type="text" value="btPret_vanz"/>	<input type="text" value="btstoc"/>
<input type="text" value="avem suma LEI"/>		<input type="text" value="de achitat"/>	<input type="text" value="lei-dupa-tranzactie"/>
<input type="text" value="btLei_final"/>		<input type="text" value="btvaloare"/>	<input type="text" value="btleidupatranzactie"/>
<input type="button" value="OK"/>	<input type="button" value="NOT OK"/>	<i>daca operatia a fost inregistrata si doriti anulara ei apasati butonul ANULARE</i>	
<input type="button" value="ANULARE"/>			

Figura S1-2: Proiectarea formularului tranzacției

Formularul de introducere a unei tranzacții are mai multe **sarcini**:

- a. *introducerea datelor despre un nou client, a tipului tranzacției (C=cumpărare, V=vânzare) a cantității de valută solicitate sau oferite;*
- b. *introducerea numelui valutei sau alegerea din valutele existente printr-un obiect de tip listă;*
- c. *afișarea informațiilor despre cursul valutar și stocul existent;*
- d. *afișarea stocului în lei înainte și după tranzacție;*
- e. *avertizarea situației de refuzare a tranzacției: la cumpărare de valută, suma în lei nu acoperă valoarea valutei oferite; la vânzare, stocul de valută existent este sub cerere;*
- e. *operarea în tabelele Valute și Tranzacții – numai dacă tranzacția poate fi efectuată;*
- f. *refacerea situației anterioare tranzacției – numai dacă se constată necesitatea anulării tranzacției după operarea ei.*

Pași în proiectarea formularului pentru adăugarea unei tranzacții

Pasul 1. Apelăm Form Design.

Pasul 2. Deschidem fereastra Data Environment, în care includem cele două tabele, astfel încât lista câmpurilor (Field List) este automat completată.

Pasul 3. Plasăm pe formular următoarele câmpuri din tabela Tranzacții: **client**, **tip_tranzacție**, **cantitate** (de tip Textbox). Observăm proprietatea Control Source.

Pasul 4. Pentru câmpul **cod_valută** – **care este cheia externă** – proiectăm un obiect de tip Combo fixând proprietățile:

```
ControlSource=tranzactii.cod_valuta;  
RowSource=valute.nume_valuta,cod_valuta;  
RowSourceType=6.
```

De fapt, beneficiind de ferestrele deschise de Combo Box Builder, vom indica mai simplu aceste proprietăți ale obiectului de control. Să nu uităm: utilitarul poate fi apelat din meniul contextual, selectând opțiunea **Build**.

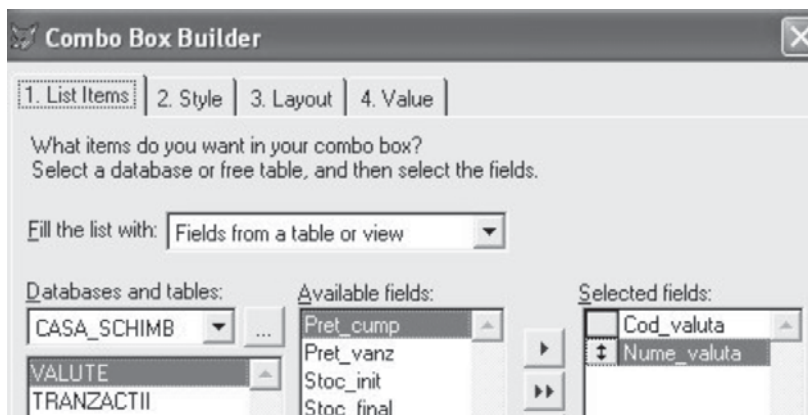


Figura S1-3: Alegerea elementelor pentru obiectul Combo

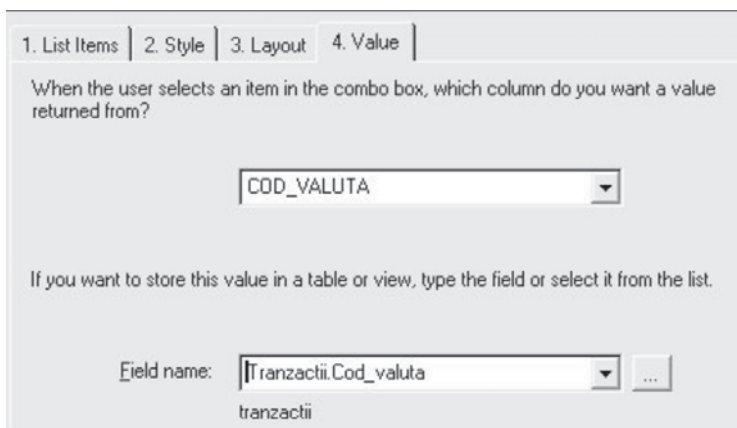


Figura S1-4: Alegerea valorii pentru obiectul Combo

Pasul 5. Plasăm pe formular câmpurile de tip textbox pentru prețul valutei și nivelul stocului final. Denumirile acestor obiecte sunt sugestive! Pentru că nu dorim ca prin operare să fie modificate valorile, folosim proprietatea `ReadOnly=.T`. Observăm schimbarea culorii asociate câmpurilor de editare. De fapt, este de preferat ca pentru operația de cumpărare să fie afișat prețul de cumpărare, iar celălalt să nu fie vizibil și invers.

De aceea, vom fixa deocamdată proprietatea `Visible=.F`. În procedura click asociată butonului `Cmccalculare` schimbăm proprietatea la valoarea `.t`. fie pentru `txtpret_cump`, fie pentru `txtpret_vanz`.

Pasul 6. Pentru `stocul2 în lei` luăm două variabile: `lei_init` și `lei_final`, pe care le memorăm în fișierul `Stoc_lei.mem`. În fereastra de comenzi dăm valori inițiale celor două variabile și salvăm cu `SAVE TO STOC_LEI`.

Pasul 7. Proiectăm caseta de text (textbox) pentru stocul final în lei și dăm ca valoare conținutul variabilei `m.lei_final`; vezi procedura click asociată butonului `Cmccalculare`.

```

CmdCalculare.clic()
RESTORE FROM STOC_LEI
thisform.txtlei_final.value=m.lei_final
*****cautare valuta si pozitionare
*****pe articolul acesteia
sele VALUTE
locate for valute.cod_valuta=;
thisform.combo1.value
thisform.refresh
*****schimbare atribut Visible
thisform.txtstoc.visible=.T.
if thisform.txttip_tranzactie.VALUE$“Cc“
*****cumparare valuta
thisform.txtpret_cump.visible=.t.
thisform.txtpret_vanz.visible=.f.

```

² Este o variantă de lucru care exemplifică și folosirea variabilelor.

```

thisform.txtvaloare.value=thisform.txtcantitate.value * thisform.txtpret_cump.value
thisform.txtleidupatranzactie.value=;
  m.lei_final-thisform.txtvaloare.value
iif thisform.txtleidupatranzactie.value<=0
  =messagebox(„atentie, nu aveti bani suficienti!“)
  endif
else
*****vanzare valuta
thisform.txtpret_cump.visible=.f.
thisform.txtpret_vanz.visible=.t.
thisform.txtvaloare.value=thisform.txtcantitate.value*thisform.txtpret_vanz.value
thisform.txtleidupatranzactie.value=;
  m.lei_final+thisform.txtvaloare.value
iif thisform.txtstoc.value<;
  thisform.txtcantitate.value
  =messagebox(„atentie, valuta existenta insuficienta!“)
  endif
endif

```

```

CmdOK.click()
*****acceptare tranzactie
RESTORE FROM STOC_LEI
M.LEI_FINAL=thisform.txtleidupatranzactie.value
IF thisform.txttip_tranzactie.value$“cC“
replace valute.stoc_final with;
  valute.stoc_final+thisform.txtcantitate.value
else
replace valute.stoc_final with;
  valute.stoc_final-thisform.txtcantitate.value
endif
replace tranzactii.valoare with;
  thisform.txtvaloare.value
SAVE TO STOC_LEI

```

```

CmdAnulare.click()
*****este situatia in care s-au facut
*****modificari in fisiere si le anulam
RESTORE FROM STOC_LEI
if m.lei_final<>thisform.txtlei_final.value
lei_final=thisform.txtlei_final.value
endif
*****
IF thisform.txttip_tranzactie.value$“cC“
replace valute.stoc_final with;
  valute.stoc_final-thisform.txtcantitate.value
else
replace valute.stoc_final with valute.stoc_final+;
  thisform.txtcantitate.value
endif
*****
select tranzactii
go bottom
delete
SAVE TO STOC_LEI

```

```
CmdNotOK.click()
SELECT TRANZACTII
GO BOTTOM
DELETE
```

```
Form1.init()
SET SAFETY OFF
set default to d:\manuale\aplicatii
use valute in 1
use tranzactii in 2
restore from stoc_lei
select tranzactii
APPEND BLANK
```

Pasul 8. Dorim să afișăm pe ecran suma în lei corespunzătoare tranzacției și proiectăm un obiect de editare cu numele `txtValoare` care va primi valoarea calculată din `cantitate*pret` (de cumpărare sau de vânzare).

Pasul 9. Va fi afișată o casetă de text, numită `txtLeiDupaTranzactie`, cu suma în lei după tranzacție, făcând diferența sau suma între stocul în lei existent la începutul tranzacției și valoarea acesteia.

Pasul 10. Proiectăm un buton de comandă, numit `cmdCalcul`, care declanșează acțiunile de afișare/calculare. Vezi codul metodei `click()`.

Pasul 11. Macheta ecran este folosită pentru o nouă înregistrare în tabela Tranzacții. Deschidem tabelele și adăugăm o nouă înregistrare prin metoda `Init` a formularului – vezi codul asociat!

Pasul 12. Proiectăm butonul de comandă care memorează operațiile din tranzacție în fișiere (butonul `cmdOK`) – vezi codul metodei `Click()`.

Pasul 13. Proiectăm un buton, numit `cmdNotOk`, pentru cazul când se renunță la tranzacție. Să nu uităm că la intrarea în formular am adăugat un articol vid. Metoda `Click()` șterge ultimul articol. Atenție! Codul tranzacției este cheie principală și nu trebuie să aibă valori vide sau duplicate. Am rezolvat acest lucru prin plasarea ca valoare implicită (Default value) funcția `Reccount()+1`. Deci, la adăugarea unui nou articol, cheia este generată automat. Dacă însă ștergem fizic articole, numărul acordat drept cheie se poate regăsi, de aceea am marcat doar cu Delete. Vă propunem ca exercițiu găsirea altei metode pentru generarea cheii primare dintr-o tabelă!

Pasul 14. Ne imaginăm situația când, din greșeală, un operator neatent a apăsă butonul `OK`. Proiectăm un buton numit `cmdAnulare`, care anulează acțiunea butonului `OK` – vezi codul metodei `cmdAnulare.Click()`.

STUDIUL DE CAZ II

Proiectarea rapidă a unei aplicații

Vom crea o aplicație care folosește toate tipurile de fișiere: tabele, vederi, interogări, rapoarte, etichete, formulare, grafice, meniuri.

Vom folosi aplicațiile wizard la realizarea acestora.

Controlul proiectării aplicației va fi exercitat prin Project Manager.

Problema

Doresc o agendă cu informații despre cunoștințele mele, prieteni, rude, colegi etc. Aș vrea să pot adăuga ușor noii mei prieteni și să pot șterge la fel de ușor pe cei cu care nu mai țin legătura; cu alte cuvinte, o aplicație care să-mi permită:

1. să mă deplasez în agendă oricum, de la început către sfârșit sau invers, dar alfabetic după numele persoanei;
2. să aflu imediat ce telefon are X;
3. să pot afișa toate cunoștințele din localitatea Y;
4. să aflu informații despre posesorul unui anumit număr de telefon Z;
5. să redactez felicitări (de Crăciun!) pentru fiecare prieten.

Etape de lucru

1. Creăm directorul `C:\alfa` unde vom depune fișierele aplicației.
2. Deschidem Project Manager, selectând **File, New, Project**.
3. Creăm o **bază de date** direct din Project Manager, astfel: în fereastra utilitarului ne poziționăm pe directorul Database, apăsăm butonul **New** și apelăm aplicația wizard. Alegem baza de date AdressBook. La ultimul pas selectăm butonul radio pentru modificarea bazei și intrăm în Database Designer. Prin meniul contextual intrăm în fereastra Table Designer și dăm un nume corespunzător câmpurilor. Atenție: trebuie schimbate și expresiile de indexare.
4. Creăm o **machetă ecran (un formular)** pentru introducerea datelor: ne poziționăm în fereastra Project Manager pe directorul Documents/ Forms și începem crearea unei machete prin clic pe butonul **Add**.

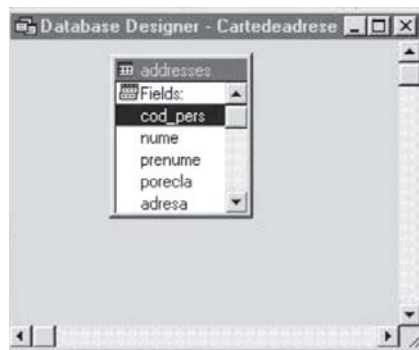


Figura S2-1: Proiectarea bazei de date



Figura S2-2: Proiectarea formularului

Alegem aplicația Form Wizard și creăm macheta. Observăm la previzualizare că etichetele asociate câmpurilor nu ne avantajează, modificăm macheta prin opțiunea corespunzătoare a ultimului ecran din Form Wizard. Salvăm cu numele **macheta_intr**.

- Realizăm un **raport** cu principalele informații din agenda de adrese. Pentru că tabela de adrese are mai multe câmpuri decât ar încăpea pe o linie a raportului, ne propunem un raport cu trei coloane, informațiile fiind dispuse pe linii.

Ne poziționăm în directorul Documents/Reports și apăsăm butonul **Add**. Folosim Report Wizard pentru un raport simplu (nici nu avem mai multe tabele!). Specificăm câmpurile, stilul tabelii și observăm la previzualizare că denumirile informațiilor au rămas cele corespunzătoare câmpurilor predefinite din tabela standard. Deci alegem opțiunea de modificare a raportului și ieșim din Report Wizard. Folosim Report Designer, care se deschide automat, pentru modificarea etichetelor asociate câmpurilor.

ADDRESSES

cod 1	cod 3	cod 5
Nume: Nancy	Nume: Janet	Nume: Steven
Prenume: Davolio	Prenume: Leverling	Prenume: Buchanan
Porecla: Paul	Porecla: Robert	Porecla:
Adresa 507 - 20th Ave. E. Apt. 2A	Adresa 722 Moss Bay Blvd.	Adresa 14 Garrett Hill
Localiate: Seattle	Localiate: Kirkland	Localiate: London
Judet: WA	Judet: WA	Judet:
Cod_postal: 98122	Cod_postal: 98033	Cod_postal: SW1 8JR
Tara: USA	Tara: USA	Tara: UK
Email nancyd@anywhere	Email janetl@anywhere.c	Email steveb@anywhere.
telefon acas a: (504) 555-9857	telefon acas a: (504) 555-3412	telefon acas a: (71) 555-4848
Telefon servici: (504) 555-9922	Telefon servici: (504) 555-9944	Telefon servici: (71) 555-5858
Fax: (504) 555-7722	Fax: (504) 555-7744	Fax:

Atenție! Modificarea textelor explicative asociate implicit de utilitar câmpurilor poate fi făcută prin următoarele manevre: 1. selectăm butonul **Label**; 2. poziționăm cursorul pe textul dorit; 3. ștergem și/sau modificăm textul explicativ. O nouă operație implică din nou cei trei pași.

Poate fi folosit și butonul **Build Lock**. Efectuarea corecțiilor se face apoi numai trecând controlul de la un obiect la altul (fără pașii 1 și 2).

- Pentru a afișa numele și adresa persoanelor cărora le trimitem felicitări, vom proiecta **etichete** prin Project Manager. Ne poziționăm pe directorul **Documents/Labels** și apăsăm butonul **New**. Alegem Label Wizard și selectăm tabela și tipul etichetei în care vom pune informațiile dorite. Numele asociat este **Felicitare.lbx**.
- Pentru a afișa persoana/persoanele care au porecla X, proiectăm o vedere (fișier view). Proiectarea **vederilor** (fișiere View) din Project Manager se face prin poziționarea pe baza de date și deschiderea utilitarului Data Designer, apoi clic pe butonul Modify al ferestrei Project Manager.

În fereastra Database Designer, deschidem meniul contextual și selectăm opțiunea **New Local View**.

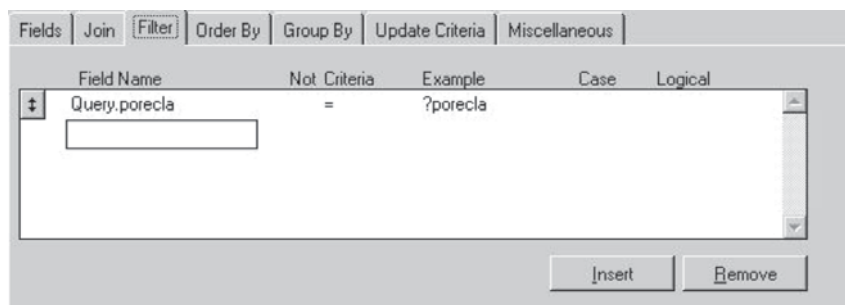


Figura S2-3: Stabilirea condiției de filtrare

Se deschide fereastra New și alegem, ca și în activitățile anterioare, View Wizard. Parcurgem pașii aplicației Wizard, iar la condiția de filtrare, în cazul nostru, punem **Adresses.porecla=?porecla**.

Atenție! Pentru ca parola să fie considerată variabilă, ea trebuie definită în fereastra View Parameters din meniul Query. Pentru aceasta, ieșim din View Wizard cu opțiunea **Save local view and modify it in View Designer**, care deschide automat View Designer. Intrăm în tab-ul Filter și vedem condiția pusă. Selectăm **Query, View Parameters** și se deschide fereastra care se precizează că variabila Parola este de tip caracter și este parametru de intrare. La rulare se cere introducerea valorii acestui parametru. Astfel este creat fișierul **Cutelefonul.vue** care selectează persoanele dintr-o anumită localitate introdusă de operator.

8. Pentru obținerea persoanelor dintr-o localitate ca **interogare** lansată din meniul aplicației vom folosi **fișierul vedere creat anterior într-o interogare**. Selectăm directorul **Data, Queries** și apăsăm butonul **Add**. Alegem Query Wizard. La primul pas, indicăm fișierul vedere pe care dorim să-l folosim, de exemplu **Dinlocalitatea**. Selectăm toate câmpurile acestei vederi, nu completăm clauza Filter și salvăm sub numele **Dinlocal.qpr**. Destinația va fi o fereastră Browse. Deci când executăm interogarea va fi solicitat numele localității și vor fi afișați într-o fereastră Browse prietenii din acea localitate.
9. Aplicația este structurată într-un **meniu**. Dorim ca o primă parte să conțină opțiuni despre aplicație și autori, a doua să lanseze programul de introducere/vizualizare prin formular. Consultările vor fi alese dintr-o listă de opțiuni: informații despre o persoană dată prin numărul ei de telefon, cine are porecla X etc.

Pentru proiectarea meniului, ne poziționăm pe directorul **Other**, subdirectorul **Menus**. Butonul **New** apelează utilitarul Menu Builder. Trecem opțiunile liniei principale. Pentru afișarea unor scurte informații, folosim editorul de texte, apoi introducem în meniu comanda de deschidere **Modify file text.txt**.

Pentru apelarea formularului, nu uităm comanda **DO FORM fis.sc**.

Operațiile de consultare vor fi alese dintr-un submeniu. Comenzile asociate sunt **DO <numefis>.qpr**. Pentru apelul raportului folosim comanda **REPORT FORM fis.frx**, iar pentru etichete, **LABEL FORM fis.lbx**. Opțiunii de iesire îi asociem comanda **SET SYSMENU TO DEFAULT**, pentru revenirea la meniul sistem.

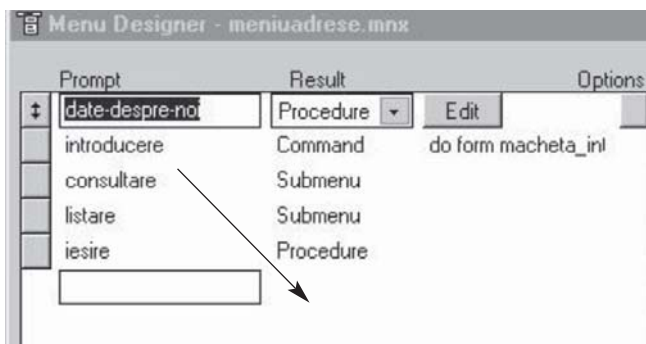


Figura S2-4: Proiectarea meniului

După proiectarea opțiunilor, nu uităm să generăm programul, selectând **Menu, Generate**. Numele meniului va fi, de exemplu, **menuadreses.mpr**.

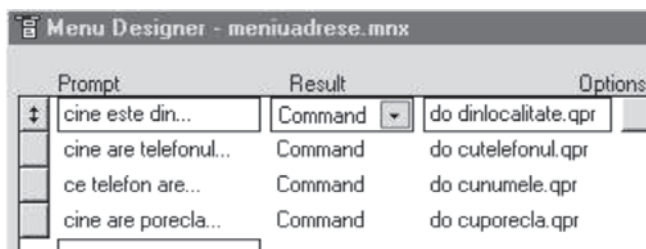


Figura S2-5: Generarea meniului

10. Proiectarea programului principal

Așa cum am arătat, este necesar să construim programul monitor: din Project Manager ne poziționăm pe directorul **Codes, Programs** și deschidem fereastra editorului de programe prin clic pe butonul **New**. Scriem următorul cod, salvat în fișierul **Princadreses.prg**.

```

oldtalk=set("talk")
set path to data\databases,data\queries,;
    document\forms,documents\reports,;
    documents\labelscodes\programs,others\menus
set talk off
localitate='iasi'
telefon='121212'
porecla='fat-frumos'
do menuadreses.mpr
read events
release all
set talk (oldtalk)
return

```

Fixăm ca fișier principal al aplicației programul **Princadreses.prg** prin clic dreapta și selectând **Set Main**.

11. Transformarea proiectului într-o aplicație se face prin butonul **Build**. Bifăm opțiunea **Recompile All Files** și selectăm **Build Application**. Numele poate fi, de exemplu, **Alfa.app**.

Salvarea se face în directorul rădăcină, acolo de unde au fost construite celelalte subdirectoare.

Lansarea în execuție a aplicației se face prin **Alfa.app**.

STUDIUL DE CAZ III

Realizarea unei aplicații executabile, generarea dischetelor de distribuție și instalarea aplicației sub Windows

Ne propunem să realizăm o aplicație executabilă pentru **gestiunea cărților** dintr-o bibliotecă. Pe parcursul manualului au fost exemplificate și probabil realizate tabelele, interogările, rapoartele și meniul principal al acestei aplicații. Programul principal este numit **Biblioteca**.

1. Creăm directorul `c:\BIBLIO`.
2. Apelăm Project Manager selectând **File, New, Project, New**. Denumim proiectul **BIB.PJX**.
3. Adăugăm la proiect toate fișierele prin clic pe butonul **ADD**.
4. Indicăm **BIBLIOTECA.PRG** ca program principal al aplicației: ne poziționăm pe linia corespunzătoare fișierului în fereastra Project Manager, executăm clic dreapta, afișăm meniul contextual și alegem **Set Main**.
5. Compilăm și grupăm toate fișierele într-un proiect prin fereastra Build Options, deschisă prin clic pe butonul Builds din Project Manager. Activăm opțiunile Recompile All Files și Display Errors, precum și butonul radio Rebuild Project.
6. Generăm aplicația executabilă prin clic pe butonul radio Build Executable. Pentru că dorim verificarea aplicației după generarea sa, activăm opțiunea Run after Build.
7. **Ne propunem să construim în continuare dischetele de distribuție a aplicației.** Folosim utilitarul Setup Wizard, lansat selectând **Tools, Wizards, Setup**.

Pasul 1: Notăm în caseta de dialog Distribution Files a primei ferestre numele directorului în care am creat aplicația: `c:\BIBLIO`.

Pasul 2: Singura opțiune pe care o selectăm este **Visual FoxPro runtime** pentru că aplicația nu folosește alte componente.

Pasul 3: Specificăm directorul imagine al kit-ului `c:\BIB`. Setăm formatul dischetei de 1,44 Mb și Web Setup.

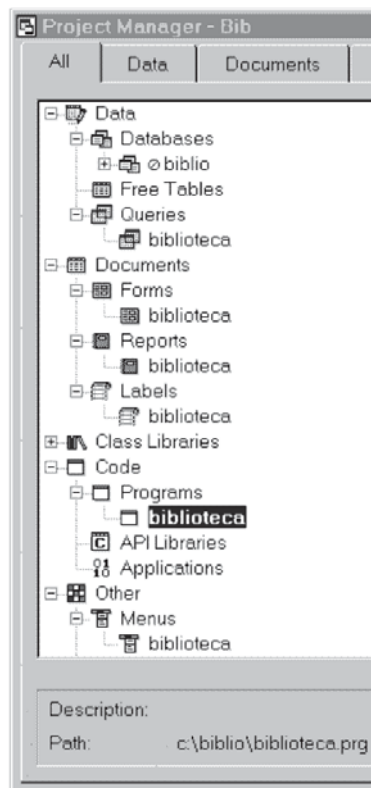


Figura S3-1:
Specificarea grupului aplicației
și asocierea unei pictograme

Pasul 4: Dăm un titlu ferestrei de dialog și trecem datele referitoare la copyright.

Pasul 5: Fixăm directorul implicit, care va fi creat pe discul destinație **C : \BIBLI**. Specificăm numele grupului pentru aplicație, care va fi adăugat la meniul Start din Windows.

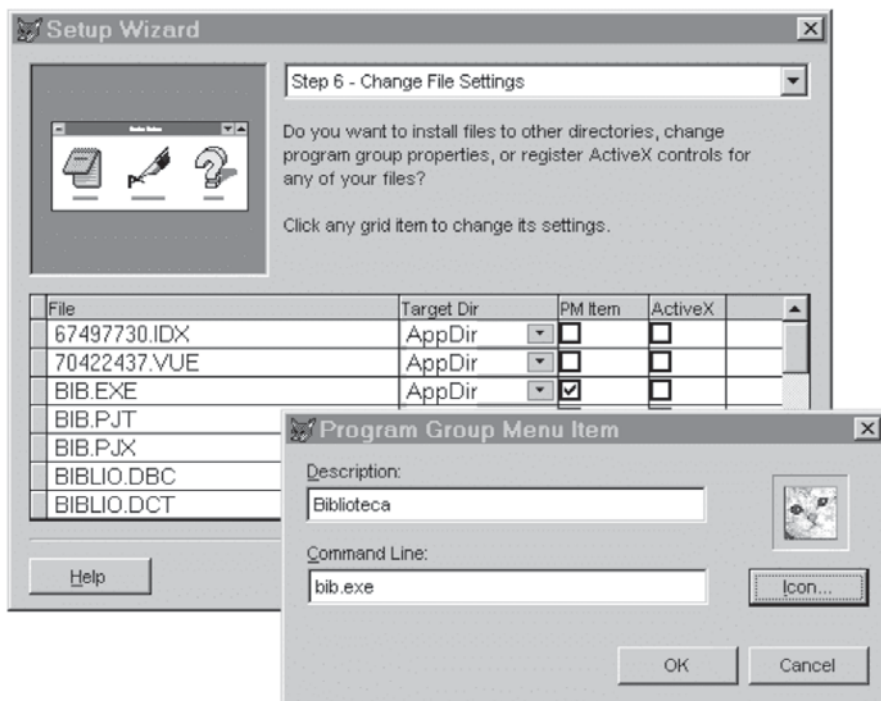


Figura S3-2: Generarea executabilelor

Pasul 6: Acceptăm ca toate fișierele să fie plasate în directorul implicit al aplicației și fixăm o pictogramă pentru fișierul **bib.exe**: selectăm în coloana PM Item comutatorul asociat acestui fișier. Se deschide fereastra Program Group Menu Item. Căutăm o pictogramă, indicăm textul asociat acesteia în caseta Description, iar în Command Line trecem instrucțiunea de apel.

Pasul 7: Activăm comutatorul **Generate a web executable file** și lansăm generatorul.

8. În vederea verificării corectitudinii operațiilor, vizualizăm directorul imagine a kit-ului **c : \BIB**. Să îl comentăm puțin:

- Directorul Websetup conține varianta pentru distribuție pe internet (Nu uitați că am setat opțiunea Web Setup în fereastra a treia a Setup Wizard!). Pentru a instala aplicația vom lansa programul **setup.exe**, aflat în acest director.

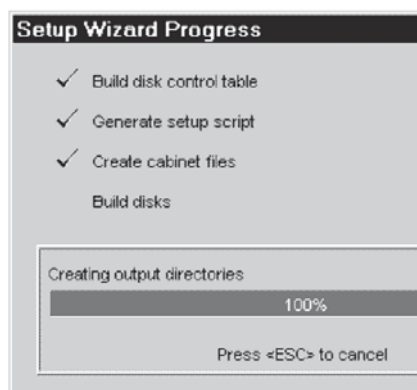
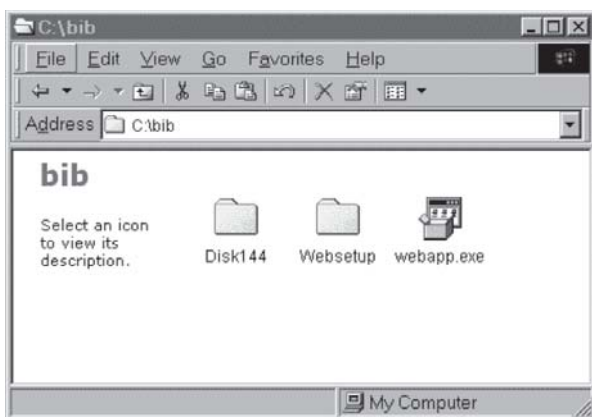


Figura S3-3: Verificarea directorului imagine a kit-ului

- Fișierul executabil **webapp.exe** este versiunea optimă a distribuției pentru internet (În fereastra a șaptea am bifat opțiunea **Generate a web executable file!**). Prin lansare în execuție, dezarchivează automat pachetul de distribuție într-un director temporar, apoi lansează același **setup.exe**.
 - Directorul Disk144 este creat pentru instalarea cu dischete. În directorul **Disk1** se află programul **setup.exe**, care va realiza instalarea aplicației.
9. Copiem conținutul directoarelor pe câte o dischetă și le predăm clientului.



10. Sfaturi pentru client:

- Copiați pe hard disc cele trei dischete și lansați aplicația **setup.exe**.
- Alegeți directorul unde va fi instalată aplicația; eventual, păstrați calea implicită, **C:\BIBLI**.

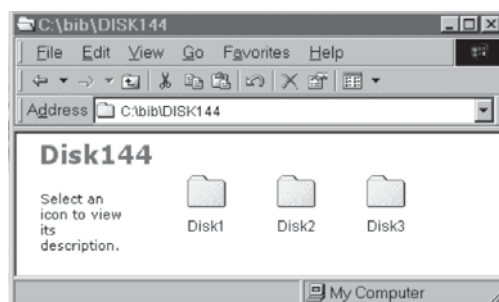


Figura S3-4: Conținutul directorului pentru instalarea cu dischete

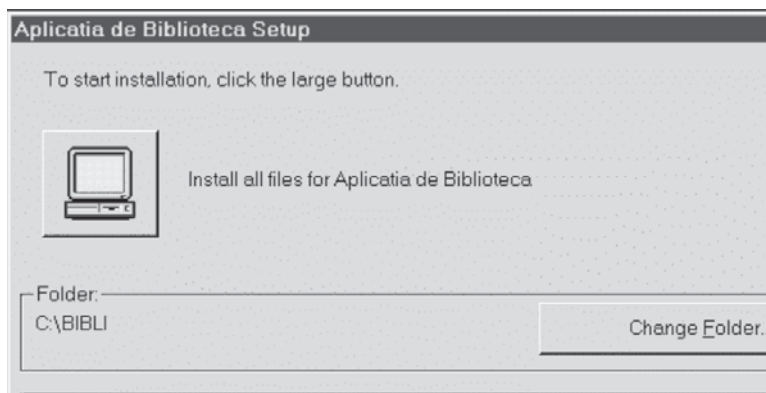


Figura S3-5: Rularea aplicației de instalare

- Verificați dacă aveți suficient spațiu și în partiția activă pentru că instalarea presupune și plasarea unor fișiere speciale (biblioteci dinamice etc.) în directorul Windows.

- Dacă doriți, schimbați numele grupului.
- Pentru începerea instalării aplicației, apăsați butonul cu pictogramă.
- Dacă totul a mers bine, apare confirmarea succesului.
- Nu vă speriați dacă apare mesajul **You must restart your computer...!** Pentru că instalarea aplicației a modificat regiștrii Windows, este necesară restartarea sistemului.
- Verificați dacă pe C: \BIBLI aveți fișierele aplicației.
- Verificați dacă în meniul Windows apare numele grupului creat la instalare.
- Căutați pictograma asociată fișierului executabil în acest grup.
- Lansați aplicația.
- Observați că nu este necesară intrarea sau ieșirea în/din mediul Visual FoxPro. Aplicația se comportă ca un program autonom.

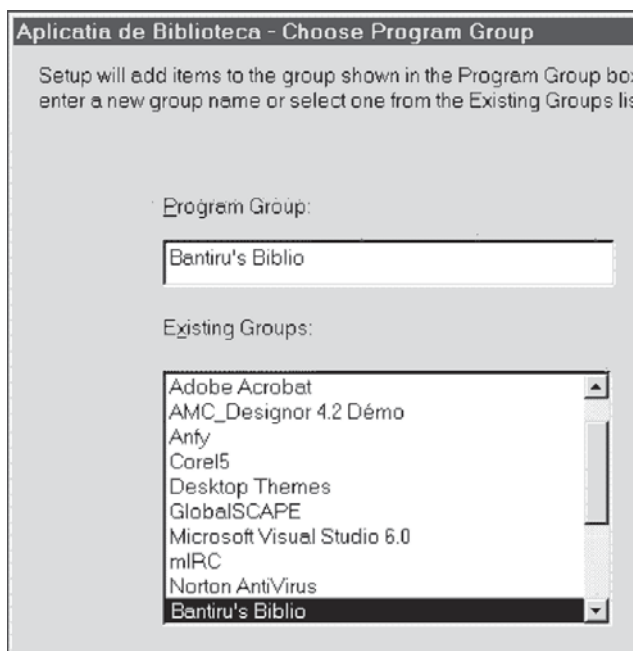


Figura S3-6: Crearea grupului aplicației la instalare

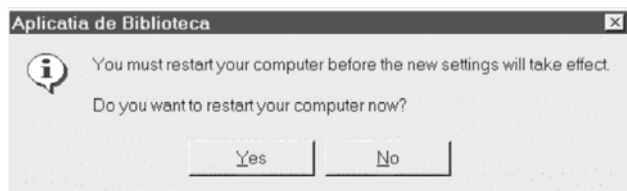


Figura S3-7: Mesaj privind necesitatea restartării după instalare

RĂSPUNSURI, COMENTARII, INDICAȚII

Pag. 29

Tabela *LECTURI* conține:

titlu	autor	editura	an_i	an_e	nume_cit	date_pers	di
c,20	c,20	c,20	n,4	n,4	c,20	m	d,8

Dar,

- Ce se întâmplă dacă ștergem articolul corespunzător unei cărți scoase din inventar? Se vor șterge și informațiile despre cititor! ==> sunt **anomalii de ștergere**
- Ce se întâmplă dacă se dorește adăugarea unui nou cititor? Se va aștepta împrumutarea unei cărți. ==> sunt **anomalii de inserare**
- Unde găsim datele despre un cititor? În toate articolele corespunzătoare cărților împrumutate și nerestituite ==> sunt **date redundante**

Deficiențele se datorează reținerii într-o singură tabelă și a informațiilor despre cărți și a celor despre cititori. Rezolvarea acestor deficiențe se realizează prin **normalizare**: împărțirea relației în alte două relații, iar fiecare să se ocupe de o temă distinctă. Obținem următoarea structură:

CARTI.DBF(cod-carte, titlu, autor, date-despre-carte)

CITITORI.DBF(cod-cititor, nume, date_pers, cod-carte, data-împrumut)

Atunci când se împrumută o carte, se adaugă un articol în tabela Cititori, iar când se restituie cartea se caută articolul și se șterge.

Dar,

- Un cititor are adresa, date personale trecute de câte ori împrumută o carte, ceea ce înseamnă **redundanță**.
- Dacă cititorul Z a împrumutat o singură carte și a restituit-o, pierdem datele personale ale acestuia!
- Nu putem afla activitatea bibliotecii printr-un indicator de felul: „numărul de cărți împrumutate într-un interval dat“ pentru că atunci când se restituie cartea, se va șterge articolul din tabela Cititori
- Nu putem afla numărul de cititori înscriși la bibliotecă pentru că fișierul CITITORI îi reține doar pe cei ce au împrumutat cel puțin o carte.
- Nu putem afla preferințele unui cititor, de câte ori a fost citită o carte etc.

Rafinăm structura, împărțind-o în alte tabele: una pentru informațiile de bază legate de cărți, alta pentru datele personale ale cititorilor bibliotecii și, în sfârșit, o tabelă cu operații. Vom face codificarea cărților printr-o informație unică folosită pentru identificare și legătură între tabele (număr de inventar).

CARTI.DBF

Cod-carte	titlu	autor	editura	editie	an_i	an_e
-----------	-------	-------	---------	--------	------	------

CITITOR.DBF

cod_cit	nume	date_pers
---------	------	-----------

OPERATII.DBF

Cod-carte	cod_cit	di	dr
-----------	---------	----	----

Pag. 32

4. *Comentarii:* Cheia relației nu poate fi decât perechea (student, sala). Dar atributul Taxa este dependent doar de sală. Nu ar exista anomalii dacă dependența ar fi totală. Observați că taxa pentru o sală este pierdută dacă se șterge articolul despre studentul care a închiriat-o. Nu putem introduce informații despre săli și taxe până când nu există un student care să închirieze respectivele săli. Se normalizează relația și se obțin relațiile SALI (nume-sală, taxa) și CHIRII (cod_student, sala).

5. *Comentarii:* Cheia este buletinul turistului (un atribut) dar, se pare că, împreună cu atributul hotel, determină atributul Taxa, ceea ce nu este corect. Spunem că Buletin determină funcțional atributul Taxa. Putem și trebuie să înlăturăm anomalia prin separarea relației în alte două relații CAZARE (turist, buletin, hotel) și TAXE (hotel, taxe/loc).

6. *Comentarii:* În exemplul acesta apar două chei posibile: (depozit, material) și (depozit, furnizor). În plus, față de cele două chei candidate, există și o dependență funcțională între furnizor și material. Anomalii de actualizare: dacă ștergem articolul al treilea, se pierde informația că „furnizorul Metalcar produce Carcase“. Se rezolvă prin spargerea relației în alte două relații: STOC (depozit, material, stoc) și FURNIZOR (furnizor, material).

Pag. 63

I.

ECHIPE (echipa C(9),grupa C(1)) cu numele și grupa fiecărei echipe

JUCATORI (nume C(9), echipa C(9), intrare D, iesire D) este un istoric cu echipe și jucători

CAMPIONAT (e1 C(9), e2 C(9), data D, loc C(9), p1 N(2), p2 N(2), arbitri M) păstrează meciurile campionatului.

Sarcini	Rezolvări
Deschideți cele trei tabele în trei zone.	<pre>use CAMPIONAT în 1 alias camp use JUCATORI în 2 alias juc use ECHIPE in 3 alias ec</pre>
Afișați în ce echipe a jucat „Popescu“ și când.	<pre>sele 2 list for nume='Popescu'</pre>
Afișați arbitrii care au fost la primele 3 meciuri.	<pre>sele camp list next 3 alte_inf</pre>
Afișați unde a arbitrat „Popescu“, la ce meciuri, când.	<pre>sele campion list e1, e2, data for "Popescu" \$ arbitri</pre>
Afișați echipele din grupa A.	<pre>sele ec list for grupa='A'</pre>
Afișați componența echipei „RAPID“ (ultima dată).	<pre>sele juc list for echipa="Rapid" and iesire={//}</pre>
Afișați localitățile unde a jucat sau va juca echipa „Rapid“.	<pre>sele camp set filter to e1="Rapid" or e2="Rapid" list loc, data, e1, e2</pre>
Afișați echipele care au jucat acasă și au pierdut.	<pre>sele 1 list e1 for p1<p2</pre>

Afișați componența echipelor care joacă la data {01.06.99}, pe stadionul „Plaișilor”? Presupunem că este un singur meci.

```
select 1
locate for data={01.06.99} .and. loc="Sportu"
sele 2
? 'echipa '+ a->e1+' este formata din :'
list nume for b->echipa=a->e1
? 'echipa '+ a->e2+' este formata din :'
list nume for b->echipa=a->e2
```

Afișați căpitanii celor două echipe care joacă în primul meci înregistrat (presupunem că prima persoană trecută în fișier la o echipă este căpitanul).

```
go top in 1
sele 2
? 'capitanul echipei '+a->e1, 'este', lookup
(b->nume, a->e1 , b->echipa)
? 'capitanul echipei '+a->e2, 'este '
?? lookup(b->nume, a->e2, b->echipa)
```

Doi prieteni fotbaliști, „Albu” și „Barbu”, se întâlnesc în campionat de două ori (tur și retur). Aflați data și locul.

```
sele b
echipa1=lookup(b->echipa, "Albu", b->nume)
echipa2=lookup(b->echipa, "Barbu", b->nume)
sele 1
list for A->e1=echipa1 .and. A->e2=echipa2 or
A->e1=echipa2 .and. A->e2=echipa1
```

Listati programul competițional de săptămâna viitoare, ordonat cronologic.

```
sele camp
sort on data to man for between(data, date(),
date()+7)
select 0
use man
list fields e1,e2,data, loc
```

Afișați meciurile, ordonate după numele echipelor care joacă acasă, pentru fiecare echipă după data desfășurării.

```
Sele camp
Sort on e1,data to man
Select 0
Use man
List
```

Pag. 110

3. Fie tabela MECI (e1 C(10), e2 C(10), g1 N(2), g2 N(2)) unde e1, e2 =codul celor două echipe, g1=numărul de goluri marcate de e1; g2=numărul de goluri marcate de e2.

&& clasam.prg

```
create dbf manevra (cod N(2), echipa C(10), puncte N(5), golaveraj N(5))
use manevra in select (0) alias man
sele meci &&parcurgem meciurile și completăm tabela manevra
scan
sele man
locate for man.cod=meci.e1
if not found()
append blank
replace cod with meci.e1
endif
repl man.puncte with man.puncte+iif(meci.g1>meci.g2,2,;
iif(meci.g1=meci.g2,1,0))
repl man.golaveraj with man.golaveraj+meci.g1-meci.g2
locate for man.cod=meci.e2
```

```

if not found()
  append blank
  replace man.cod with meci.e2
endif
repl man.puncte with man.puncte+iif(meci.g2>meci.g1,2,;
  iif(meci.g1=meci.g2,1,0))
repl man.golaveraj with man.golaveraj+meci.g2-meci.g1
sele meci
endscan
sele man
index on str(punctaj)+str(golaveraj) tag v
List

```

4. Concurenti (nume C(10),grupa C(3),nr N(2),ts C(8),tp C(8))

Concurs(grupa C(3), start C(5), interval N(2))

1) Procedura de citire a concurenților

```

sele concurenti
clear
accept 'nume' to m.nume
accept 'data nasterii' to dn
accept 'SEX' to sex
varsta=year(date())-year(dn)
replace grupa with
  sex+ltrim(str(varsta))
replace nume with m.nume

```

2) Procedura de codificare a concurenților (câmpul Nr)

Varianta 1:

```

use concurenti in 1
index on grupa tag xx unique
copy to man field grupa && sunt
  grupele distincte
index on grupa+nume tag x
use man in 2
sele 2
scan
sele 1
locate for b.grupa=a.grupa
n=0
do while found()
n=n+1
repl nr with n
continue
enddo
sele 2
endscan

```

Varianta 2:

```

use concurs in 1
use concurenti in 2 order nume
sele 1
scan
sele 2
copy to man for a.grupa=b.grupa
delete all for a.grupa=b.grupa
pack
sele 3
use man
repl all nr with recno()
use
sele 2
append from man
erase man.dbf
sele 1
endscan

```

Varianta 3:

```

use concurenti
index on grupa+nume tag xx
do while not eof()
x=grupa
i=0
do while grupa=x
i=i+1
repl nr with i
skip
enddo
enddo

```

Varianta 4:

```

use concurenti in 1
index on grupa tag xx unique
copy to array A
index on grupa+nume tag x
for i=1 to alen(A,1)
n=0
locate for grupa=A[i,1]
do while found()
n=n+1
replace nr with m.n
continue
enddo
endfor

```

3) Procedura de completare a momentului de start pentru fiecare concurent

Varianta 1:

```
use concurs in 1
use concurenti in 2
sele 2
index on nume tag nume
sele 1
scan
sele 2
count to y for b.grupa=a.grupa
if y#0
copy to array x for
    b.grupa=a.grupa
delete for b.grupa=a.grupa
pack
x[1,4]=a.start
yy=a.interval
for i=2 to alen(x,1)
x[i,4]=timp(x[i-1,4],yy)
endfor
append from array x
release x
endif
sele 1
endscan
return
```

```
function timp
param t,d
o=val(left(t,2))
m=val(right(t,2))+d
o=o+int(m/60)
m=mod(m,60)
return
    str(o,2)+' ':'+str(m,2)
```

Varianta 2:

```
use concurs in 1 order grupa
use concurenti in 2
sele 2
set relation to grupa into a
repl all tp with timp( a.start,
    a.interval*(b.nr -1))
return
```

Varianta 3:

```
use concurs in 1
use concurenti in 2
sele 1
scan
sele 2
locate for b.grupa=a.grupa
    t1=a.start
    t2=0
do while found()
repl tp with t1
t1=timp(t1, a.interval)
continue
enddo
sele 1
endscan
return
```

4) Procedura de afișare a câștigătorilor

procedure afis

```
sele 0
create table man (grupa C(3),
    nr N(3), durata N(6))
append from concurenti
sele 1
use concurenti
sele 3
use man
set relation to recno() into a
repl all c.durata with;
    diferenta(a.ts,a.tp)
index on grupa tag gr
list c.grupa,a.nr,a.tp,a.ts,c.durata
sort on durata to man2
use man2
set relation to recno() into a
index on grupa tag x unique
list c.grupa, a.numa, a.tp, a.ts,
    c.durata
use
erase man.dbf
erase man2.dbf
return
```

5) Procedura de trecere a momentului de sosire

procedure sosire

```
input' codul?' to x
accept' grupa?' to y
select concurenti
locate for nr=x and grupa=y
if found()
replace ts with time()
else
wait 'atentie coduri eronate'
endif
return
function diferenta
param ts,tp
os=val(left(ts,2))
ms=val(right(ts,2))
mins=os*60+ms
op=val(left(tp,2))
mp=val(right(tp,2))
minp=op*60+mp
return mins-minp
```

Pag. 114. 1. Concurs de frumusețe

```
*programul principal *
set talk off
public array p[10]
if not f1()
&& validare date
wait 'date incomplete'
return
endif
do p1
&&
input 'criteriul ? ' to x
? 'punctajul maxim obtinut la;
  criteriul ',x,',' , f2(x)
do p2
  && marcare persoane ce nu intrunesc
  && minimul cerut la un criteriu
do p3
&& lista persoanelor
  castigatoare
&& pe criterii

sir=f3()
&& criteriile unde sunt mai multi
&& castigatori

if not empty(sir)
? 'sunt mai multe persoane;
  castigatoare la criteriile '+sir
else
? ' nu sunt criterii la care sa;
  existe mai multe persoane cu;
  acelasi punctaj maxim'
endif
nr=f4()
&& numarul persoanelor
  necalificate
&& la un criteriu si cu maxim la
&& alt criteriu
if nr=0
? 'nu sunt persoane cu punctaje;
  extreme'
else
for i=1 to nr
? pers[i]
endfor
endif

? 'clasamentul competitiei'
do p4
&& punctajul total pe concurenta.
&& se va lua in considerare
&& existenta aceleiasi valori la
  mai
&& mute concurente

close all
erase total.dbf
return
```

```
function f1
&&verificare completitudine date
public array p[10]
if not (file('conc.dbf') and;
  file('minime.mem') and;
  file('rezult.dbf'))
wait 'date de plecare incomplete'
result=.F.
else
  restore from minime
  additive
  select 1
  use rezult
&& verificare corectitudine fisier
&& rezult
  nr criterii=alen(p)
  select 2
  use conc
  nr_candidat=reccount(2)
  use
  sele rezult
  index on nume+str(criteriu);
  tag n unique
  count to z
  if z # nr_candidat*nr_criterii
  rezult=.f.
  else
  rezult=.T.
  endif
  set order to 0
&& fisierul rezult este lasat
&& neordonat
  delete tag n
  && stergem reperul index
  anterior
&& creat
endif
return rezult

function f2
&& punctajul maxim la un criteriu
dat
&& ca parametru
parameters x
calculate max(punctaj) for;
  criteriu=x to y
return y

procedure p3
  && lista castigatorilor pe
  criterii
select rezult
set heading off
for i=1 to alen(p)
ma=f2(i)
?'rezultatul cel mai bun la;
  criteriul ', i
list nume, punctaj for
  criteriu=i;
  and punctaj=ma
endfor
set heading on
return
```

```

procedure p1
&& trece sub forma de tabel
&& rezultatele
select result
index on nume tag nume unique
count to g
nr_criterii=alen(p)
dimension a[g, nr_criterii+1]
copy to array a field nume
delete tag nume
index on nume tag nume
&& fara selectie unica
text    && varianta de rezolvare
  scan
  poz=ascan(a, a.nume,1)
  a[asubscript(a,poz,1),
  criteriu+1]=punctaj
  endscan
  endtext && sfarsit varianta

for i=1 to alen(a,1)
locate for nume= a[i,1]
do while found()
a[i,criteriu+1] =punctaj
continue
enddo
endifor
*afisare tablou*
for i=1 to alen(a,1)
?
for j=1 to alen(a,2)
?? a[i,j], ' '
endifor
endifor
return

```

```

procedure p2
&& marcare persoane care nu au
&& punctajul de calificare
select result
recall all
delete all for
  punctaj<p[criteriu]
copy to manevra for deleted()
select 2
use manevra
scan
sele 1
delete for b.nume=a.nume
sele 2
endscan
use
erase manevra.dbf
sele 1
index on nume tag nume unique
list for deleted()
set order to 0
return
function f3
  && criteriile unde sunt mai
  multi
  && castigatori

sir=''
for i=1 to alen(p)
ma=f2(i)
count for criteriu=i and;
  punctaj=ma to zz
if zz>1
sir=sir+iiif(empty(sir),str(i,3),
  ','+str(i,3))
endif
endifor
return sir

```

```

function f4
&& determina persoanele
  necalificate
&& la un criteriu si castigatoare
&& la altul
public array pers[20]
nr_caz=0
select result
index on nume tag nume
do while not eof()
  x=nume
  store .f. to bun, rau
do while not eof() and nume=x
  rau=deleted()
  if punctaj = f2(criteriu)
  bun=.T.
  endif

```

```

procedure p4
&& lista primilor trei clasati*
sele result
index on nume tag nume for;
  not deleted()
total on nume field punctaj to
  total
sele 2
use total
index on punctaj tag pct1;
  descending unique
copy to man_unic
index on punctaj tag pct2 desc
sele 3
use man_unic
i=0
scan next 3

```

```

skip
if eof()
exit
endif
enddo
if bun and rau
nr_caz=nr_caz+1
pers[nr_caz]=x
endif
enddo
return nr_caz

```

```

i=i+1
? 'premiul ', i
sele 2
list nume for b.punctaj=c.punctaj
sele 3
endscan
sele 3
use
erase man_unic.dbf
return

```

Pag. 147

1. Calculul necesarului de aprovizionat

procedure calcul

```

create dbf nec (mat n(1), nt n(5), na
n(5))
use stoc in 1
use plan in 2
use consum in 3
use nec in 4
sele 3
index on mat tag mat
do while not eof(3)
x=mat
sum c.cant*cautplan(c.prod) while;
c.mat=x to y
do adauga_art with x, y
enddo
sele 4
repl all na with nt-cautstoc(d.mat)
list
return

```

Varianta 1:

Se va crea un fișier NEC cu necesarul total (câmpul NT) și necesarul de aprovizionat (câmpul NA) pentru fiecare cod-material (câmpul MAT).
 Ordonăm fișierul CONSUM pe materiale.
 Pentru fiecare material i:
 facem suma produselor dintre cantitatea unitar necesară pentru produsul j și planul pentru produsul j
 generăm un articol în fișierul NEC
 Completăm câmpul NA
 Afișare

function cautplan

```

param cod
sele 2
locate for b.prod=cod
if found()
rez=b.cant
else
rez=0
endif
sele 3
return rez

```

procedure adauga_art

```

param x, y
sele 4
append blank
repl mat with x,nt;
with y
sele 3
return

```

function cautstoc

```

param x
sele 1
locate for a.mat=x
sele 4
return
iif(found(1),a.cant,0)

```

```

use stoc in 1 order mat
use plan in 2 order prod
sele 3
use consum
copy to man
use man
repl all c.cant with;
  c.cant * cautplan(c.prod)
index on mat tag mat
total on mat to nec field cant
use nec
repl all prod with;
  c.cant - cautstoc(c.mat)
?' material, necesar total,;
  necesar de aprovizionat'
set heading off
list c.mat, c.cant, c.prod
close all
erase man.dbf
return

```

function cautplan

```

param x
sele 2
seek x
sele 3
return iif(found(2), b.cant, 0)

```

Varianta 2:

1. Se folosește o tabelă de manevră (copie a fișierului CONSUM) înlocuind câmpul CANT cu produsul dintre cantitatea necesară și planul pentru produsul corespunzător.
2. Se face totalizarea pentru același material în fișierul TOT.
3. Se poate completa necesarul de aprovizionat în câmpul PROD (atenție la lungimea codului de produs!) prin diferența dintre necesarul total (aflat în câmpul CANT) și stocul din materialul corespunzător.
4. Se afișează.

function cautstoc

```

param x
sele 3
return iif(seek(x, 1), a.cant, 0)

```

```

sele 2
use plan
index on prod tag prod
sele 3
use consum
set relation to prod into b
sele 1
create table nec (mat N(5),;
  cant N(5), na N(5))
append from stoc
repl all a.na with suma(a.mat)
?' material, necesar total, necesar;
  de aprovizionat'
list mat, na, na-cant
return
function suma
param x
sele 3
sum c.cant*b.cant for c.mat=x to y
sele 1
return y

```

Varianta 3:

Se folosește relația dintre fișierul CONSUM și PLAN pe atributul PROD. Se presupune că fișierul STOC are toate codurile de materiale chiar dacă stocul este zero. Pentru fiecare cod material din stocuri se scade suma produselor consum unitar și plan

Varianta 4:

Putem organiza datele în tablouri și vom nota:

1. Tabloul consum specific $C(i,j)$ conține: cantitatea necesară din materialul i pentru realizarea unei unități din produsul j ($i = 1, n$ produse, $j = 1, m$ materiale);
2. Vectorul plan $P(i)$ = cantitatea planificată a fi realizată din produsul i ;

3. Vectorul stocuri S(J) = stocul existent din materialul j (j = 1,m).

	consum specific material / produs	plan producție	necesar materiale	stocuri	de aprovizionat
	p1	p2	p3	*	
m1	10	5	25	=	p1 100
m2	0	10	10	=	p2 100
				=	p3 200
				=	m1 6500
				=	m2 3500
				=	m1 500
				=	m2 1000
				=	m1 6000
				=	m2 2500

- Presupunem că fișierul STOC nu are toate codurile de materiale, ci numai cele cu stoc pozitiv, fișierul Plan, de asemenea, nu conține decât acele coduri de produse care se fabrică la un moment dat.
- Pentru dimensionarea tablourilor vom determina atât codul de valoare maximă pentru materiale, cât și pentru produse din fișierele corespunzătoare.
- După calculare, se trec rezultatele în baza de date NEC(mat, nt, na), unde mat=codul materialelor, nt=necesar total, na= necesar de aprovizionat.

&& program principal

```

nmat=det_cod(1)
nprod=det_cod(2)
dimension
  cs(nmat,nprod),P(nprod),s(nmat),;
nt(nmat), na(nmat)
store 0 to cs, s, p, nt, na
do incarca with 1
do incarca with 2
do incarca with 3
do calcul_nt && Nt= CS*P
do afis with nt && afisare vector NT
do calcul_na && NA=NT-S
do afis with na && afisare vector NA

```

&&continuare

```

sele 1
create dbf nec (mat N(5),nt;
N(5),na N(5))
for i=1 to nmat
append blank
repl mat with i,nt with
nt[i],;
na with na[i]
endfor
delete for nt=0
pack &&articole
close all &&nefolosite
return

```

function det_cod

```

param cod
sele 1
use consum
calculate max(mat),
max(prod) to xm, xp
if cod=1
use stoc
calculate max(mat) to ym
use
return max(xm, ym)
else
use plan
calculate max(prod) to yp
use
return max(xp, yp)
endif
endif

```

procedure incarca

```

param x
do case
case x=1
use consum
scan for mat#0 and
prod#0
CS[mat, prod]=cant
endscan
case x=2
use plan
scan for prod#0
P[prod]=cant
endscan
otherwise
use stoc
scan for mat#0
S[mat]=cant
endscan
endcase
return

```

procedure calcul_nt

```

for i=1 to nmat
nt[i]=0
for j=1 to nprod
nt[i]=nt[i]+cs[i, j]*P[j]
endfor
endfor
return

```

procedure calcul_na

```

for i=1 to nmat
NA[i]=NT[i]-S[i]
endfor
return

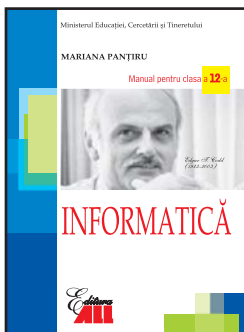
```

procedure afis

```

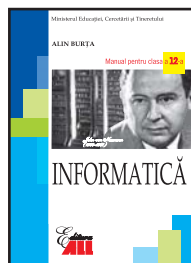
param v
?
for i=1 to alen(v)
if v[i]>0
?? str(v[i], 5)
endif
endfor
return

```



INFORMATICĂ
Manual pentru clasa a 12-a

filiera teoretică / profil real /
specializarea: matematică-informatică;
filiera vocațională / profil militar MApN /
specializarea: matematică-informatică



INFORMATICĂ
clasa a 12-a
ALIN BURȚA