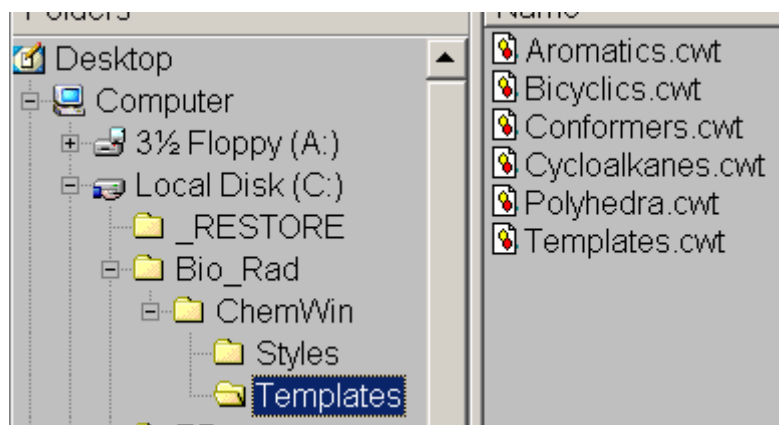


1. Baze de date și SGBD

Calculatoarele au fost folosite încă din 1950 pentru *stocarea și procesarea datelor*. Un deziderat major al *sistemelor informatice* este de a realiza produse software care să localizeze eficient datele pe suportul fizic și să-l încarce rapid în memoria internă pentru procesare. La baza unui sistem informatic se află un *set de fișiere* memorate permanent pe unul sau mai multe suporturi fizice.



Gama largă de aplicații ale informaticii necesită acces rapid la mari cantități de date. Iată câteva exemple:

- sistemele computerizate de marcare din supermarketuri trebuie să traverseze întreaga linie de produse din magazin;
- sistemele de rezervare a locurilor la liniile aeriene sunt folosite în mai multe locuri simultan pentru a plasa pasageri la numeroase zboruri la date diferite;
- calculatoarele din biblioteci stochează milioane de intrări și accesează citații din sute de publicații;
- sistemele de procesare a tranzacțiilor în bănci și casele de brokeraj păstrează conturi care generează fluxul mondial de capital;
- motoarele de căutare World Wide Web scanează sute de pagini Web pentru a produce răspunsuri cantitative la interogări aproape instantaneu;
- sute de mici întreprinzători și organizații utilizează bazele de date pentru a stoca orice de la inventare și personal la secvențe ADN și informații despre obiecte provenite din săpături arheologice.

Un produs software care presupune managementul fișierelor suportă descompunerea logică a unui fișier în *înregistrări*. Fiecare înregistrare descrie o entitate și constă dintr-un

Crearea și exploatarea bazelor de date relaționale

număr de *câmpuri*, unde fiecare câmp dă valori unei anumite proprietăți (sau atribut) al entității.

	Last_name	First_name	Acct_nbr	Address_1	City
▶	Davis	Jennifer	1023495.0000	100 Cranberry St.	Wellesley
	Jones	Arthur	2094056.0000	10 Hunnewell St	Los Altos
	Parker	Debra	1209395.0000	74 South St	Atherton
	Sawyer	Dave	3094095.0000	101 Oakland St	Los Altos
	White	Cindy	1024034.0000	1 Wentworth Dr	Los Altos

Un fișier simplu cu înregistrări este adecvat pentru date comerciale cu complexitate redusă, cum ar fi inventarul dintr-un magazin sau o colecție de conturi curente pentru clienți.

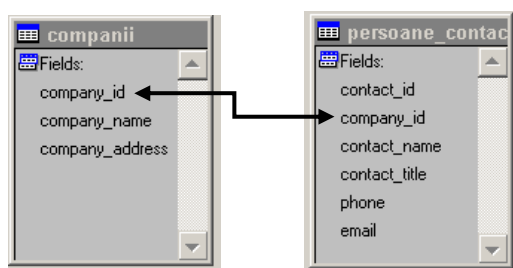
Un *index* al unui fișier constă dintr-o *listă de identificatori* (care disting înregistrările) *împreună cu adresele înregistrărilor*. De exemplu numele poate fi folosit pentru a identifica înregistrările unor persoane. Deoarece indexurile pot fi mari ele sunt uzual structurate într-o formă ierarhică și sunt navigate cu ajutorul pointerilor. Formele ierarhice arborescente sunt frecvent folosite datorită vitezei mari de traversare.

Problemele reale ale procesării datelor solicită frecvent legarea datelor din mai multe fișiere. Astfel, în mod natural s-au conceput structuri de date și programe de manipulare a datelor care să suporte legarea înregistrărilor din fișiere diferite.

3 modele de baze de date au fost create pentru a suporta legarea înregistrărilor de tipuri diferite:

- *modelul ierarhic*: tipurile înregistrărilor sunt legate într-o structură arborescentă (de exemplu înregistrările unor angajați s-ar putea grupa după o înregistrare care să descrie departamentele în care aceștia lucrează); IMS (Information Management System produs de IBM) este un exemplu de astfel de sistem;
- *modelul rețea*: se pot crea legături arbitrare între diferitele tipuri de înregistrări (de exemplu înregistrările unor angajați s-ar putea lega pe de o parte de o înregistrare care să descrie departamentele în care aceștia lucrează și pe de altă parte supervizorii acestora care sunt de asemenea angajați);
- *modelul relațional*: în care toate datele sunt reprezentate într-o formă tabelată simplă.

În modelul relațional descrierea unei entități particulare este dată de setul valorilor atributelor, stocate sub forma unei linii în tabel și numită relație. Această legare a n valori de attribute furnizează cea mai potrivită descriere a entităților din lumea reală.



Crearea și exploatarea bazelor de date relaționale

Modelul relațional suportă *interogări* (cereri de informații) care implică mai multe tabele prin asigurarea unor legături între tabele (operația *join*) care combină înregistrări cu valori identice ale unor atribute ale acestora.

Statele de plată, de exemplu, pot fi stocate într-un tabel iar datele personalului beneficiar în altul. Informațiile complete pentru un angajat pot fi obținute prin reunirea acestor tabele (*join*) pe baza numărului personal de identificare.

Pentru a suporta o varietate de astfel de structuri de baze de date, o largă varietate a software denumită *sistem de gestiune a bazelor de date* este necesară pentru a stoca și reda datele și pentru a pune la dispoziția utilizatorului posibilitatea de a interoga și actualiza baza de date.

În 1970, Ted Codd (IBM, părintele SQL), nemulțumit de performanțele COBOL și IMS formulează principiul de lucru al bazelor de date relaționale. Codd afirmă că SGBD trebuie să recunoască comenzi simple și trebuie să fie aproape de utilizatori prin punerea împreună a comenzilor potrivite pentru găsirea a ceea ce se dorește. Ceea ce Codd numește model relațional se bazează pe două puncte cheie:

- să furnizeze un mod de descriere a datelor cu numai cu structura lor naturală, ceea ce înseamnă că trebuie realizat acest lucru fără impunerea nici unei structuri adiționale pentru scopuri de reprezentare în calculator;
- de asemenea, să furnizeze baza pentru un limbaj de date de nivel înalt care va conduce la o maximă independență între programe, pe de o parte, și reprezentarea în calculator, pe de altă parte.

Cu alte cuvinte modelul relațional consistă din:

- independența datelor față de hardware și modul de memorare;
- navigarea automată sau un limbaj de nivel înalt neprocedural pentru accesarea datelor;

În loc ca să se proceseze câte o înregistrare, programatorul utilizează limbajul pentru a specifica operații unice care trebuie realizate asupra întregului set de date.

Limbajele de generația a 4-a (4th GLs) sunt mai aproape de limbajul uman ca limbajele de nivel înalt (de generația a 3-a, 3th GLs). Primele dintre acestea sunt FOCUS (Information Builders) SQL (IBM), QBE (Query by example, IBM), dBASE (succesorul lui SQL).

Necesitatea pentru mai multă flexibilitate și performanță din partea modelelor de date cum ar fi de a suporta aplicațiile științifice sau ingineresti a făcut ca să se extindă conceptul de model relațional așa încât intrările în tabele să nu mai fie simple valori ci să poată fi programe, texte, date nestructurate mari în formă binară sau orice alt format solicitat de utilizator. Un alt progres s-a făcut prin încorporarea conceptului de *obiect* devenit esențial în limbajele de programare. În bazele de *date orientate obiect* toate datele sunt obiecte. Obiecte

Crearea și exploatarea bazelor de date relaționale

se pot lega între ele printr-o *relație de apartenență* pentru a forma o familie mai largă și mai diversă de obiecte (în anii '90 au fost lansate primele sisteme de management orientat obiect OODMS). Datele care descriu un transport pot fi stocate, de exemplu, ca familie mai largă care poate conține automobile, vapoare, vagoane, avioane. Clasele de obiecte pot forma *ierarhii* în care obiecte individuale pot moșteni proprietăți de la obiectele situate deasupra în ierarhie. Bazele de date multimedia, în care vocea, muzica și informația video se stochează împreună cu informațiile de tip text, devin tot mai frecvente și își imprimă trendul în dezvoltarea sistemelor de gestiune a bazelor de date orientate obiect.

O secvență tipică pentru un limbaj 4th GL este:

FIND ALL RECORDS WHERE NAME IS "TUCKER"

SQL (Structured Query Language) este un limbaj standard industrial pentru crearea, actualizarea și interogarea sistemelor de management ale bazelor de date relaționale.

Prima versiune standardizată a SQL a apărut în 1986 și conține construcțiile de bază ale limbajului pentru definirea și manipularea tabelor de date. O revizie în 1989 a adăugat limbajului extensii pentru integritatea referențială și generalizează constrângerile de integritate. O altă extensie în 1992 furnizează facilități în manipularea schemelor și administrarea datelor și de asemenea substanțiale îmbunătățiri în ceea ce privește definirea și manipularea datelor. Dezvoltarea sistemului este în desfășurare pentru a face din acesta un limbaj computațional complet pentru definirea și managementul obiectelor complexe persistente. Aceasta include generalizarea și specializarea ierarhiilor, moștenire multiplă, tipuri de dată utilizator, generatoare și construcții declarative, suport pentru sistemele bazate pe cunoștințe, expresii interogative recursive și instrumente adiționale de administrare a datelor. Include de asemenea tipuri abstracte de date, identificatori de obiecte, metode, moștenire, polimorfism, încapsulare și toate celelalte facilități care sunt asociate uzual cu managementul datelor de tip obiect.

În prezent, industria bazelor de date, ca segment al industriei de software generează anual aproximativ 8 miliarde de dolari. Companiile care dețin supremația pe acest segment de piață sunt IBM, Oracle, Informix, Sybase, Teradata (NCR), Microsoft, Borland.

2. Formele Backus-Naur

BNF (Backus-Naur Form, numite originar Backus Normal Form și redenumite apoi la sugestia lui Donald Knuth) formează o metasintaxă utilizată pentru a exprima gramatici independente de context. BNF este unul dintre cele mai utilizate notații metasintactice pentru specificarea sintaxei limbajelor de programare și seturile de comenzi ale acestora. Pentru detalii suplimentare vezi „<http://src.doc.ic.ac.uk/computing/internet/rfc/rfc2234.txt>”.

Crearea și exploatarea bazelor de date relaționale

Fie o formă BNF a unei adrese poștale din U.S.:

```
<postal-address> ::= <name-part> <street-address> <zip-part>
<personal-part> ::= <name> | <initial> "."
<name-part> ::= <personal-part> <last-name> [<jr-part>] <EOL>
                | <personal-part> <name-part>
<street-address> ::= [<apt>] <house-num> <street-name> <EOL>
<zip-part> ::= <town-name> ", " <state-code> <ZIP-code> <EOL>
```

Aceasta se traduce prin: „O adresă poștală constă dintr-o parte de nume, urmată de o parte de adresă stradală și urmată de o parte de cod poștal. O parte personală constă din prenume sau dintr-o inițială urmată de un punct. O parte de nume constă din următoarele: o parte personală urmată de nume urmat de un opțional <jr-part> (Jr., Sr., sau numărul dinastiei) și sfârșit de linie sau o parte personală urmată de o parte de nume (aceasta ilustrează recursivitatea în formele BN, acoperind cazul persoanelor care folosesc mai multe nume sau prenume și/sau inițiale). O adresă stradală constă dintr-un specificator opțional de apartament, urmat de număr și numele străzii. Partea de cod poștal constă din numele orașului, urmat de virgulă, urmat de codul statului și orașului urmat de sfârșit de linie.”

De observat că multe lucruri (cum ar fi formatul părții personale, specificatorul de apartament sau de codul orașului au rămas nespecificate. Aceste detalii lexicale sunt presupuse evidente din context sau sunt specificate în altă parte.

Sunt multe variante și extensii ale BNF, de exemplu prin introducerea wildcardurilor ? și *. Două dintre acestea sunt EBNF și ABNF.

3. Baze de date relaționale

Fie un simplu exemplu de carte de adrese, nimic prea complex, doar ceva care memorează nume, adrese, numere de telefon, emailuri și atât. Să memorăm acum astfel de informații într-un fișier text cu delimitatori. Dacă prima linie servește ca cap de tabel și virgula este folosită ca separator, acest fișier ar putea arăta ca mai jos:

```
Name, Addr1, Addr2, City, State, Zip, Phone, E-mail
Jay Greenspan, 211 Some St, Apt 2, San Francisco, CA, 94107,
4155551212, jgreen_1@yahoo.com
Brad Bulger, 411 Some St, Apt 6, San Francisco, CA, 94109,
4155552222, bbulger@yahoo.com
John Doe, 444 Madison Ave, , New York, NY, 11234, 2125556666,
nobody@hotmail.com
```

Nu este mult de văzut la acesta, însă este cel puțin încărcabil în calculator. Utilizând orice limbaj de programare (din generația a 3-a) se poate scrie un cod care să deschidă acest fișier și să preia informația. Oricum am pune problema însă în implementarea acestui cod, se

Crearea și exploatarea bazelor de date relaționale

arată a fi o bună bucată de cod de scris. Dacă se dorește ca această informație să se poată ordona și interoga după o varietate de criterii, care de exemplu să sorteze alfabetic după nume sau să găsească toți oamenii care au o anumită valoare a codului de localitate, este într-adevăr dureros. Ne putem lovi acum de o altă problemă majoră dacă datele dorim să fie utilizate în cadrul unei rețele de o mulțime de utilizatori, cum ar fi modificările pe care le-ar efectua un utilizator în timp ce alt utilizator se poziționează pe o înregistrare mai jos în fișier. Probabil va trebui să blocăm la scriere fișierul atunci când mai mulți utilizatori îl accesează.

Este evident deci că soluția memorării acestuia sub formă de fișier text nu a fost fericită. Este nevoie de *un sistem de stocare care să reducă cantitatea de prelucrări și accesări* ale informației din fișier de către programul responsabil cu gestiunea acestuia. Un tabel simplu ca tabelul 1 ar trebui să funcționeze tocmai bine:

Tabelul 1. Tabel simplu pentru stocarea datelor

name	addr1	addr2	city	state	zip	phone	e-mail
Jay Greenspan	211 Some St	Apt 2	San Francisco	CA	94107	4155551212	jgreen_1@yahoo.com
Brad Bulger	411 Some St	Apt 6	San Francisco	CA	94109	4155552222	bbulger@yahoo.com
John Doe	444 Madison Ave		New York	NY	11234	2125556666	nobody@hotmail.com

Acum acesta este aproape convenabil. Este ușor de ajuns la sfârșit și de verificat dacă vreun program accesează acest tabel. Este ușor de accesat o linie din acest tabel odată fără a afecta pe ceilalți. În acest sens, dacă 2 sau mai mulți utilizatori doresc să insereze informații în această tabelă ei nu se vor suprapune în acțiunea lor. Dacă se va dori extragerea unor informații din tabel, cum ar fi toate persoanele care sunt din California, nu va fi necesar să se prelucreze și ordoneze fișierul. Un program care ar opera pe această tabelă va trebui doar să rezolve următoarea subproblemă: *afișează toate liniile la care conținutul coloanei **State** este egal cu 'CA'*. Da, este frumos, însă nu este destul.

Obiectivul Dr. Codd, părintele SQL, a fost de a avea un model al informației care să nu creeze anomalii. Se pot identifica 3 situații de anomalie: la Actualizare, Ștergere și Inserare.

Să presupunem că o structură tabelată poate rapid și ușor gestiona cereri multiple și să analizeze ce se întâmplă dacă informația devine ceva mai complexă, cum ar fi cazul ilustrat în tabelul 2:

Tabelul 2. Tabelă cu stocare problematică

id	company_name	company_address	contact_name	contact_title	phone	email
1	BigCo Company	1121 43 rd St	Jay Greenspan	Vice President	4155551212	jgreen_1@yahoo.com
2	BigCo Company	1121 43 rd St	Brad Bulger	President	4155552222	bbulger@yahoo.com
3	LittleCo Company	4444 44 th St	John Doe	Lackey	2125556666	nobody@hotmail.com

Crearea și exploatarea bazelor de date relaționale

Ce se întâmplă dacă de exemplu, firma BigCo se hotărăște să-și schimbe sediul? Va trebui să actualizăm adresa sa în două linii. Poate fi și o sursă de erori dacă modificarea se face manual în fiecare linie și cineva introduce greșit una din cele două noi adrese. Rezultă deci că o cale mai bună de a manipula aceste date este de a lua numele companiei și adresa acesteia și a le pune într-o tabelă separată. Rezultatul poate fi ca în tabelele 3 și 4.

Tabelul 3. Companii

company_id	company_name	company_address
1	BigCo Company	1121 43 rd St
2	LittleCo Company	4444 44 th St

Tabelul 4. Persoane de contact

contact_id	company_id	contact_name	contact_title	phone	email
1	1	Jay Greenspan	Vice President	4155551212	jgreen_1@yahoo.com
2	1	Brad Bulger	President	4155552222	bbulger@yahoo.com
3	2	John Doe	Lackey	2125556666	nobody@hotmail.com

Ceea ce s-a realizat în acest caz prin separarea celor două categorii de informații și introducerea unei coloane de legătură (*company_id*) este că s-a creat o *relație între cele două tabele* și de aici vine numele de bază de date relațională.

Deși avem exact aceleași informații ca la început, totuși, există o diferență, faptul că tocmai le-am segmentat. Putem acum schimba adresa atât pentru Jay cât și pentru Brad prin modificarea unei singure linii. Acesta este un fapt convenabil, oricum.

Să presupunem că se întâmplă ca d-nul Doe să fie șters din baza de date în forma din tabelul 2. Dar se poate întâmpla ca cineva să vrea să ceară lista tuturor companiilor cu care ai avut contact anul trecut. În forma curentă, ștergându-l pe Doe, vom șterge și informația despre companie odată cu el. Această problemă se numește ștergerea anormală.

Tabelul 2. Tabelă cu ștergere anormală

id	company_name	company_address	contact_name	contact_title	phone	email
1	BigCo Company	1121 43 rd St	Jay Greenspan	Vice President	4155551212	jgreen_1@yahoo.com
2	BigCo Company	1121 43 rd St	Brad Bulger	President	4155552222	bbulger@yahoo.com
3	LittleCo Company	4444 44 th St	John Doe	Lackey	2125556666	nobody@hotmail.com

Dacă însă se păstrează structura din tabelele 3 și 4, se poate face ștergerea numai din tabela 4, și înregistrarea cu compania poate să rămână în tabela 3, așa încât în acest caz problema ștergerii anormale nu mai există.

Privind din nou datele din tabelul 2 putem observa că scopul principal al acestei tabele este de a stoca contacte și nu companii. Situația devine paradoxală atunci când dorim să

Crearea și exploatarea bazelor de date relaționale

inserăm o companie dar nu și persoana de contact. De cele mai multe ori, ar trebui să așteptăm până când avem date specifice de contact pentru ca să putem adăuga în baza de date. Este evident o restricție ridicolă.

4. Microsoft Visual FoxPro

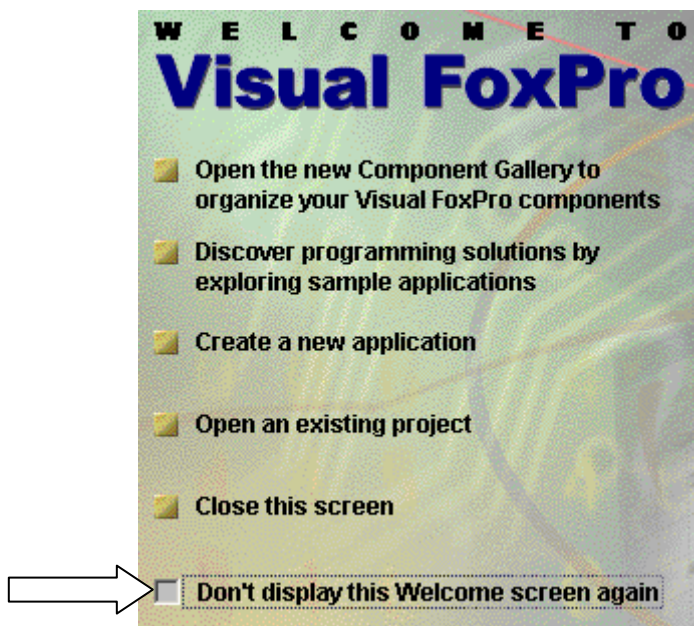
Microsoft Visual FoxPro face parte din pachetul Microsoft Visual Studio distribuit de firma Microsoft (TM).

Începând cu versiunea 6.0, Microsoft Visual Studio este un pachet integrat, care conține Visual Basic, Visual C++, Visual FoxPro, InterDev, Visual J++, SourceSafe și o bogată documentație denumită MSDN (Microsoft Developer Network Library).



Lansarea aplicației în execuție se poate face din Taskbar, Start→Programs→Microsoft Visual Studio 6.0→Microsoft Visual FoxPro 6.0 sau dacă a fost instalat cu opțiunile implicite de instalare din locația: C:\Program Files\Microsoft Visual Studio\Vfp98\VFP6.EXE".

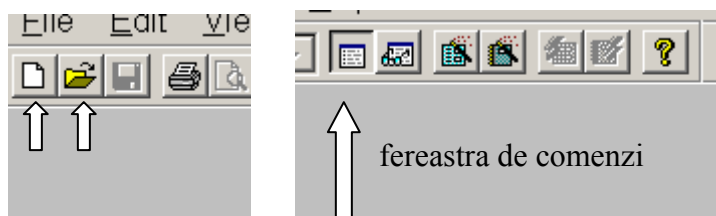
La pornire se activează un Wizard opțional:



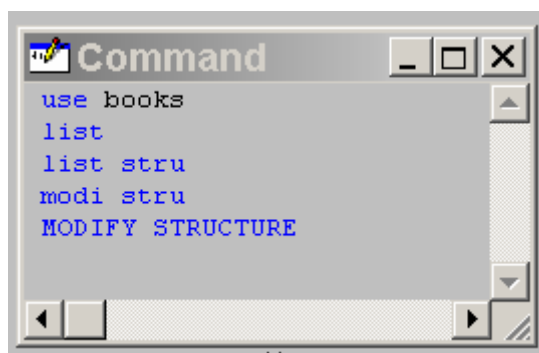
care se poate dezactiva prin marcarea Checkbox-ului.

Pentru a ajunge la meniul aplicației se alege opțiunea **Close this screen**. Cu ajutorul barei de instrumente se pot crea sau deschide baze de date, tabele, interogări, forme, rapoarte, etichete, programe, clase sau proiecte. Oricare din operațiile efectuate cu ajutorul barei de instrumente se pot efectua și din fereastra de comenzi; ea se activează/dezactivează ca în fig.

Crearea și exploatarea bazelor de date relaționale

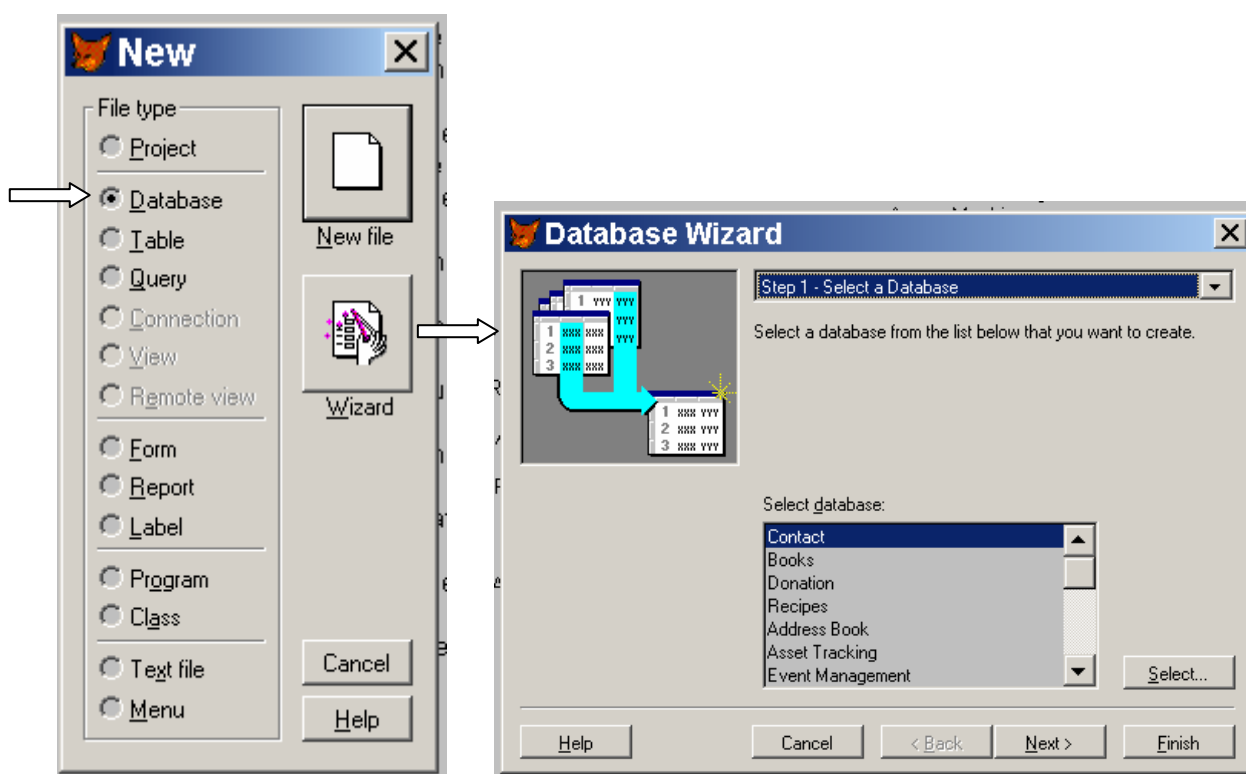


Operațiile asupra bazelor de date create pe care le efectuăm din bara de instrumente sunt oricum înregistrate în fereastra de comenzi, aceasta păstrând istoria activității sesiunii de lucru curente:



5. Crearea unei baze de date

Cu ajutorul butonului New se activează o fereastră cu butoane radio ca în figură:

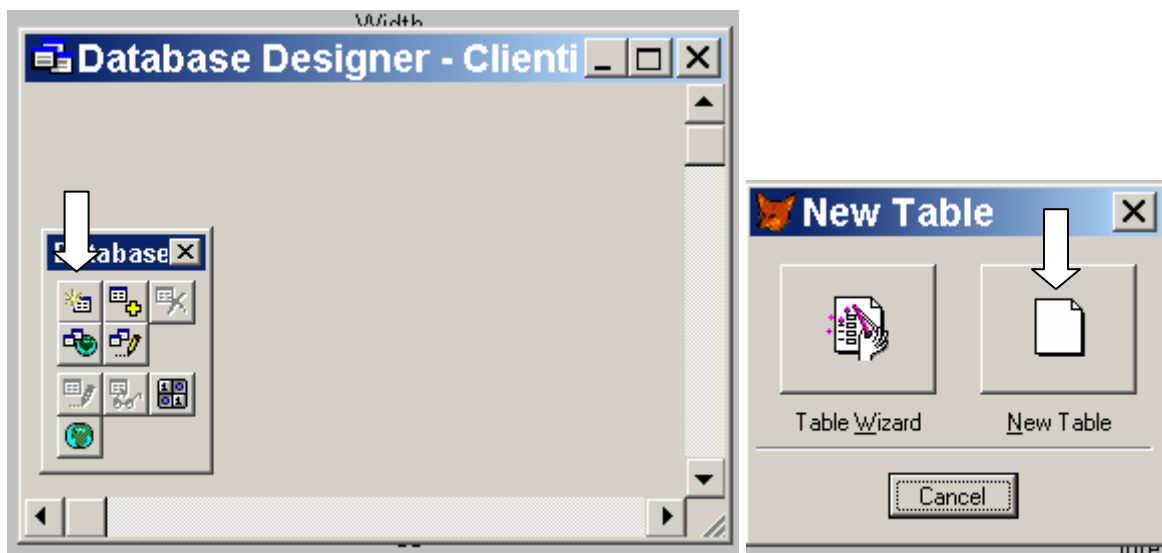


Se poate alege a se crea o bază de date cu ajutorul wizardului ca în figură în care într-o succesiune de 5 pași se precizează caracteristicile noii baza de date care se dorește să se creeze.

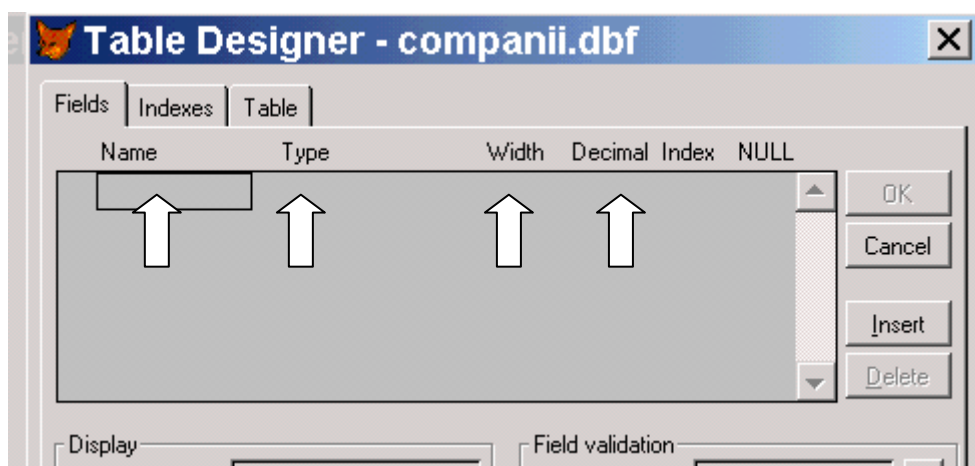
Crearea și exploatarea bazelor de date relaționale

Baza de date așa cum s-a putut constata în exemplul prezentat anterior este compusă din mai multe tabele. Să presupunem că vrem să memorăm informațiile prezentate în tabelele 3 și 4. În acest caz, putem alege să construim o nouă bază de date pe baza creării unui nou fișier în care apoi să adăugăm cele două tabele, și anume COMPANII și PERSOANE_CONTACT.

Alegem deci New/Database, New File, introducem numele noii baze de date (să spunem CLIENTI când se generează următorul rezultat:

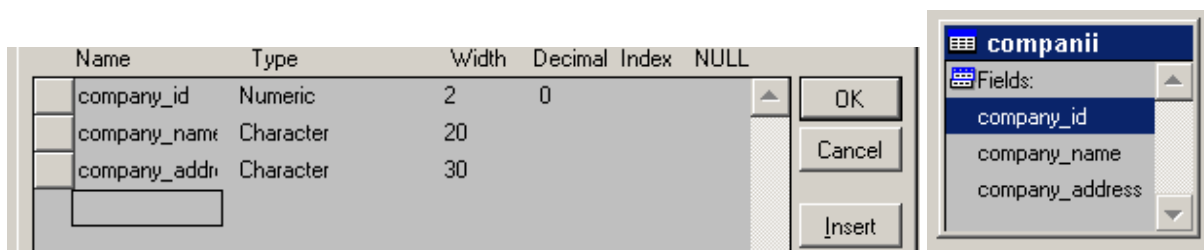


În Această coală avem posibilitatea să construim structura celor două tabele (COMPANII și PERSOANE_CONTACT). Activarea consecutivă a butoanelor New Table și apoi din nou New Table urmat de introducerea numelui primei tabele (COMPANII) duce la activarea constructorului de tabele:



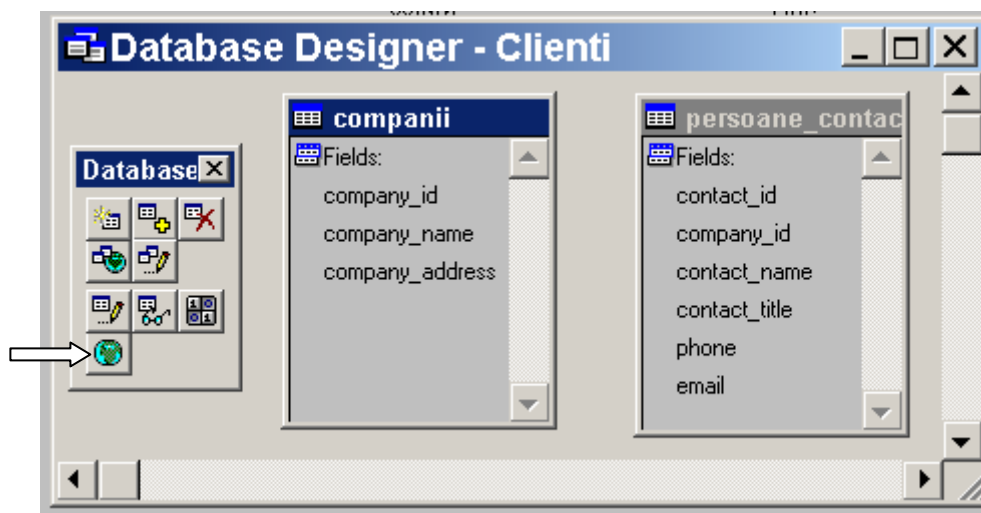
Se precizează succesiv în această tabelă numele, tipul, lungimea și precizia zecimală acolo unde este cazul pentru fiecare câmp al bazei de date, până când se obține un rezultat ca în figura următoare:

Crearea și exploatarea bazelor de date relaționale



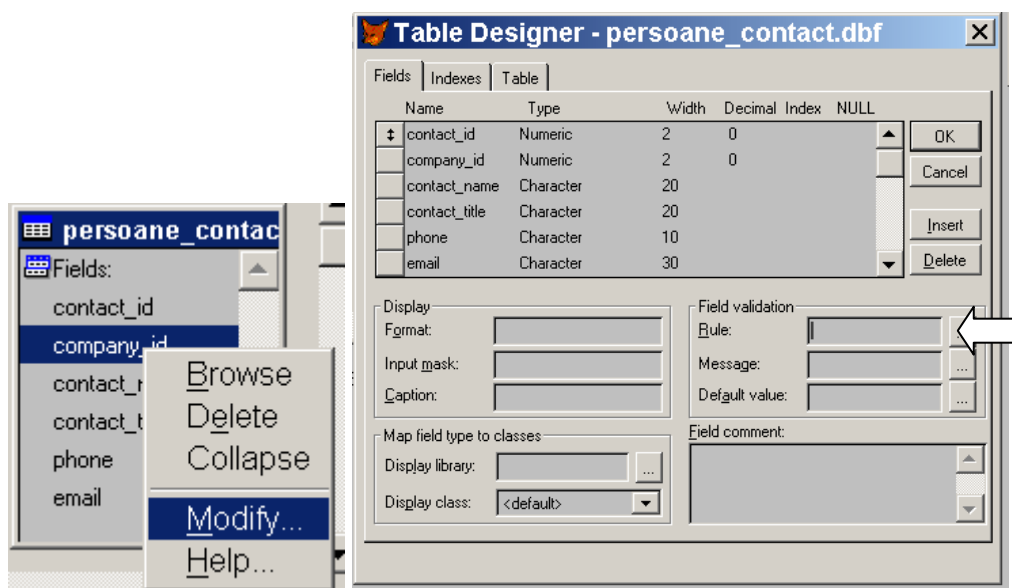
Se va apăsa butonul Ok și în acest moment se va adăuga automat pe foaia de lucru a bazei de date desenul din dreapta care reprezintă tabela creată.

Se repetă procedura și pentru cea de-a doua bază de date, când se obține:

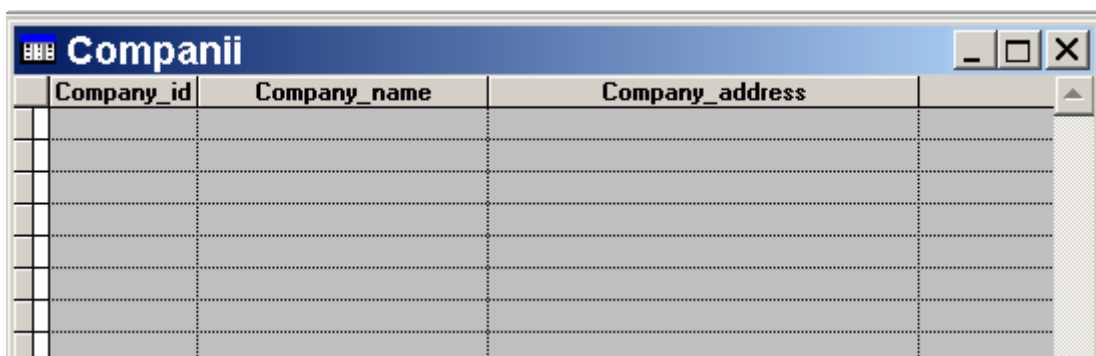


Se poate stabili acum o relație de validare la completare. Se activează cu click dreapta ca în figură proprietățile câmpului 'contact_id' din 'persoane_contact' și din fereastra de modify se stabilește regula:

Companii.company_id > 0



Se completează apoi bazele de date cu informațiile necesare prin dublu click asupra tabelelor, când se activează ferestre de tip browse în care se adaugă câte o înregistrare cu CTRL+Y sau Table/Append New Record:



Company_id	Company_name	Company_address

După aceasta oricând ne putem întoarce la foaia tabelului pentru a vizualiza conținutul tabelului sau pentru a face adăugări sau modificări.

6. Normalizarea unei baze de date

Așa cum s-a putut vedea în exemplul prezentat, crearea unei baze de date relaționale presupune identificarea tuturor relațiilor între atributele entităților care sunt stocate sau se vor stoca în baza de date. Se poate ca aceste relații să se identifice după ce structura bazei de date a fost creată. Oricum, procesul prin care se elimină cele 3 anomalii (la modificare, la ștergere și la adăugare) se numește *normalizare*. Înțelegerea normalizării este vitală pentru lucrul cu baze de date relaționale.

Normalizarea nu este un proces cu care se începe sau se termină designul bazei de date. Este un proces care se aplică oricând se identifică anomalii. Experiența și instinctul totdeauna joacă un rol important în crearea unei bune baze de date.

Normalizarea se poate realiza prin trecerea succesivă a datelor prin câteva *forme normale*. Până în prezent s-au stabilit 7 astfel de forme normale, dintre care primele 3 asigură o calitate destul de bună a organizării relaționale a bazei de date și majoritatea bazelor de date relaționale sunt organizate până la această formă.

Prima formă normală a datelor necesită ca:

- datele să fie structurate într-un tabel;
- fiecare coloană trebuie să conțină o singură valoare de un singur tip, adică să existe o singură valoare în fiecare celulă; nu sunt permise șiruri sau alte forme de reprezentare a mai mult de o valoare pe celulă;
- fiecare coloană trebuie să aibă un nume unic;
- tabelul trebuie să aibă un set de valori care identifică în mod unic o linie; valorile din această coloană se numesc *chei primare* pentru tabel;
- nu trebuie să existe două linii identice în tabel;
- nu sunt permise grupuri repetitive de date;

Ultima afirmație necesită explicații. Fie datele din tabelul 5:

Tabelul 5. Tabel cu grupuri repetitive de date

company_id	company_name	company_address	contact_name	contact_title	phone	email
1	BigCo Company	1121 43 rd St	Jay Greenspan	Vice President	4155551212	jgreen_1@yahoo.com
2	BigCo Company	1121 43 rd St	Brad Bulger	President	4155552222	bbulger@yahoo.com
3	LittleCo Company	4444 44 th St	John Doe	Lackey	2125556666	nobody@hotmail.com

Cum se observă, zona marcată conține informații identice. Ea formează un grup repetitiv. După ce vom înlătura aceste coloane și le plasăm în propriul lor tabel se ajunge la prima formă normală.

Cheile primare sunt o coloană sau un set de coloane care au pe fiecare linie o valoare unică în șirul valorilor coloanei respective. În tabelul 5 se poate vedea cum s-a inclus o astfel de coloană (*company_id*). Toate browserele de baze de date posedă un instrument de a defini o astfel de coloană. De exemplu, în MySQL aceasta se numește câmp *auto_increment*. Pot însă fi *chei primare* seriile de buletin, adresele email sau URL-urile. Singura condiție este ca datele să fie unice.

De exemplu, dacă informațiile de contact dintr-o astfel de agendă de adrese presupun memorarea de informații pentru companii cu mai multe reprezentanțe, probabil cea mai bună soluție este de a memora identificatorul reprezentanței și adresa într-un tabel separat în care *company_id* și eventual *company_city* să formeze cheia primară.

Dependența datelor este un element esențial în organizarea lor relațională. O coloană este *dependentă* (de cheia primară) dacă ea nu poate exista în tabel când cheia primară este înlăturată.

A doua formă normală intervine când la sfârșitul primei normalizări obținem o cheie primară formată din mai multe coloane. Să presupunem că în urma procesului de separare a tabelului de adrese (v. tabelul 5) obținem o tabelă în forma:

Tabelul 6. Tabel care nu e în forma normală 2

company_name	company_location	company_ceo	company_address
BigCo Company	San Francisco	Bill Hurt	1121 43 rd St
LittleCo Company	LA	LittleCo Company	4444 44 th st

Aici, *company_name* și *company_location* formează cheia primară multiplă.

O *adăugare anormală* se produce atunci când dorim să adăugăm o nouă adresă pentru BigCo Co. Vom avea numele CEO (Chief Executive Officer) repetat în linia adăugată.

Transformăm acest tabel în a doua formă normală prin eliminarea liniilor care sunt doar parțial dependente de cheia primară. CEO este dependent doar de coloana *company_name* și nu este dependent de coloana *company_location*. Pentru a ajunge la a 2-a

Crearea și exploatarea bazelor de date relaționale

formă normală, se mută liniile care sunt doar parțial dependente de cheia primară multi-coloană într-un tabel propriu.

A doua formă normală nu se aplică pentru tabelele care au o cheie primară formată de o singură coloană.

A **3-a formă normală** rezolvă *dependențele tranzitive*. O *dependență tranzitivă* este atunci când o coloană există în tabel dar nu este condiționată direct de cheia primară. În loc de aceasta, ea este condiționată de alte câmpuri, care la rândul lor sunt condiționate de cheia primară.

O cale rapidă de a se ajunge la a 3-a formă normală este de a ne uita la toate câmpurile din tabel și de a ne întreba dacă aceste câmpuri descriu cheia primară. Dacă nu, locul lor nu este aici (în această tabelă). Dacă se dorește ca agenda să memoreze mai multe informații de contact, ca în tabela următoare:

Tabelul 7. Tabel care nu e în a 3-a formă normală

contact_id	contact_name	contact_phone	assistant_name	assistant_phone
1	Bill Jones	4155555555	John Bills	2025554444
2	Carol Shaw	2015556666	Shawn Carlo	6505556666

atunci se pune problema dacă nu cumva prin adăugarea acestora nu se alterează normalizarea acestora. Este posibil, chiar probabil ca un asistent să deservească mai multe reprezentanțe, ceea ce face ca *assistant_name* și *assistant_phone* să apară în tabel mai mult decât odată. Acestea vor forma atunci un *grup repetitiv*, care deja a fost discutat cum se elimină.

7. Tipuri de relații

Este esențial a se crea un grup de tabele care să nu aibă anomalii. Acestea totdeauna includ coloane care mențin (definesc) relațiile între aceste tabele. Sunt 3 tipuri de relații în domeniul bazelor de date:

- *relații 1 la n*: de departe cel mai frecvent caz, când o valoare dintr-o coloană referă mai multe câmpuri într-un alt tabel:

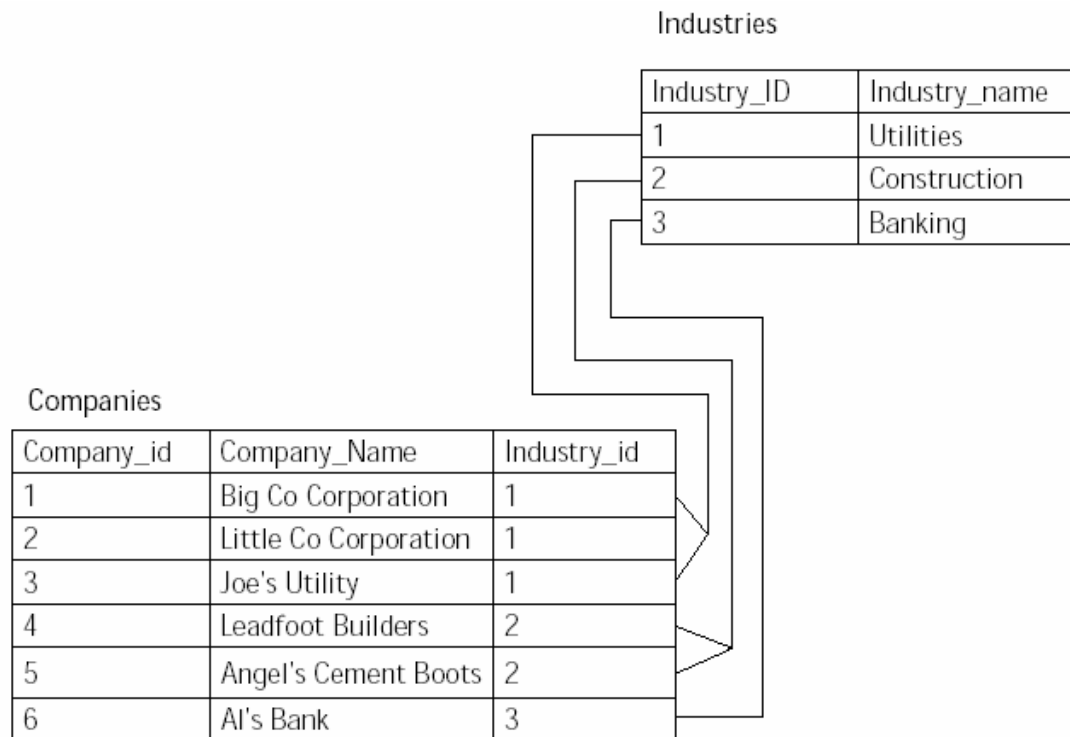
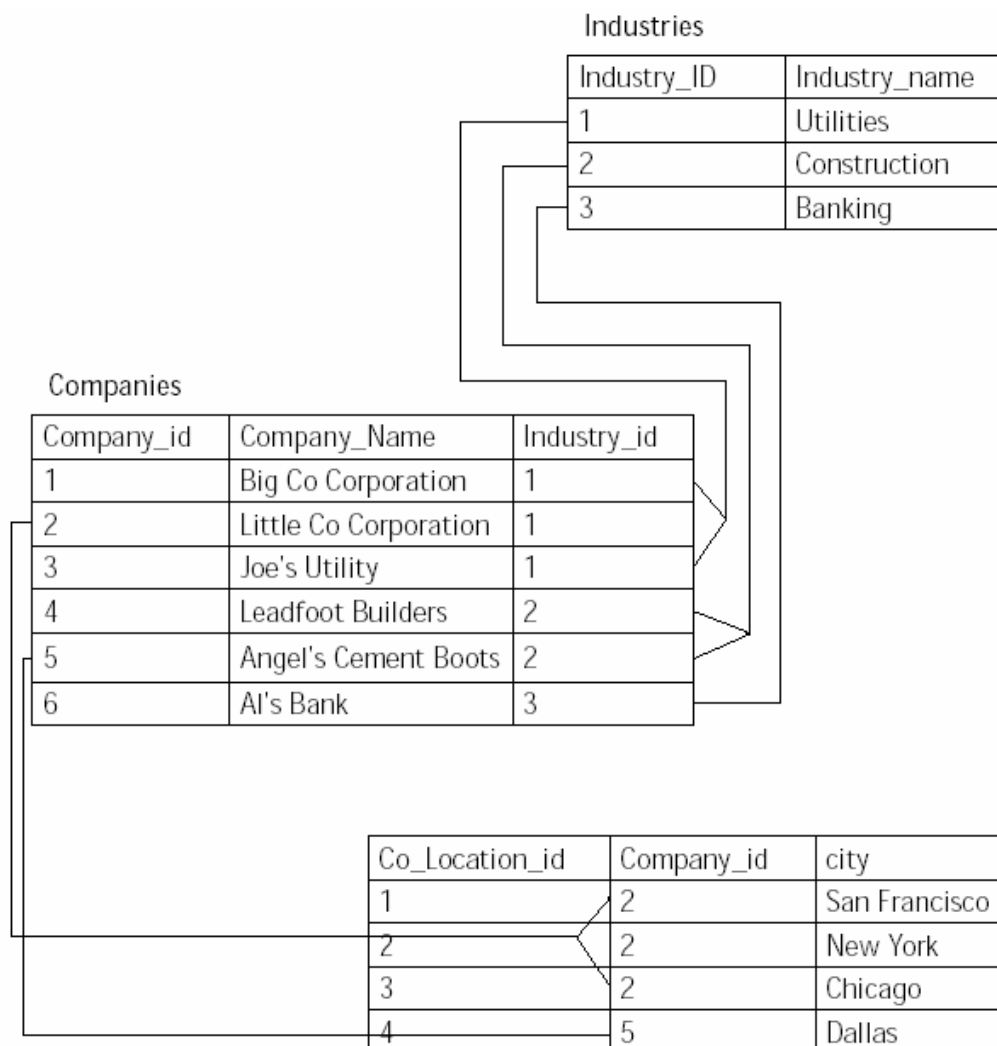
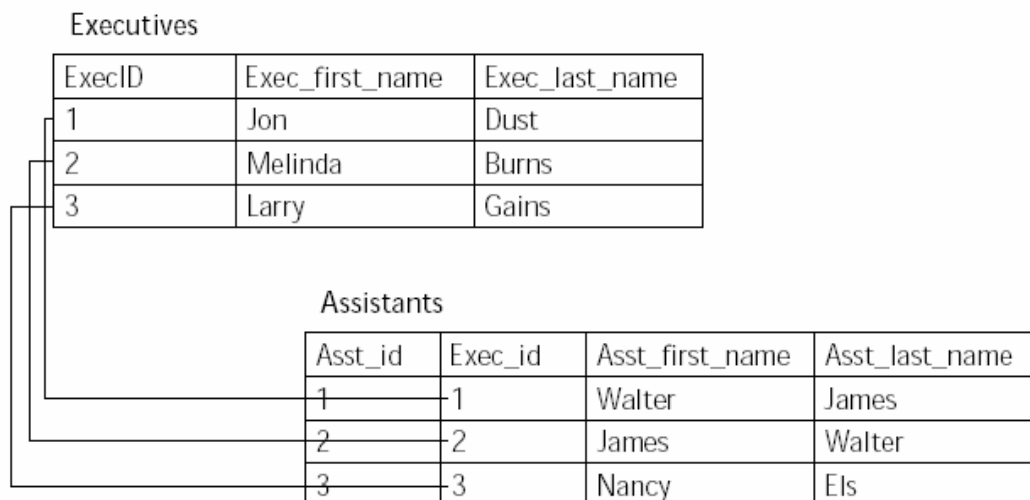


Figura următoare arată cum tabela de Companii poate fi unul din capetele încă unei relații 1 la n cu tabela reprezentanțelor:



Crearea și exploatarea bazelor de date relaționale

- *relații 1 la 1*: sunt de fapt relații 1 la n în care o linie dintr-un tabel este legată cu o linie dintr-un alt tabel; ele rezultă de obicei în urma proceselor de normalizare, așa cum ar putea rezulta în urma separării datelor din tabelul 7:



- *relații m la n*: acestea lucrează diferit față de cele două anterioare; să presupunem că compania păstrează informații despre o varietate de publicații care pot fi distribuite persoanelor de contact, în funcție de cererea acestora; pentru început, se creează tabela care va memora tipurile de publicații:

Tabelul 8. Tabela publicațiilor

newsletter_id	newsletter_name
1	Weekly
2	Monthly
3	Bi-monthly
4	Annual

Se poate acum adăuga o coloană în tabelul de persoane de contact pentru a preciza publicațiile care se vor trimite, ca în tabelul 9:

Tabelul 9. Persoane de contact

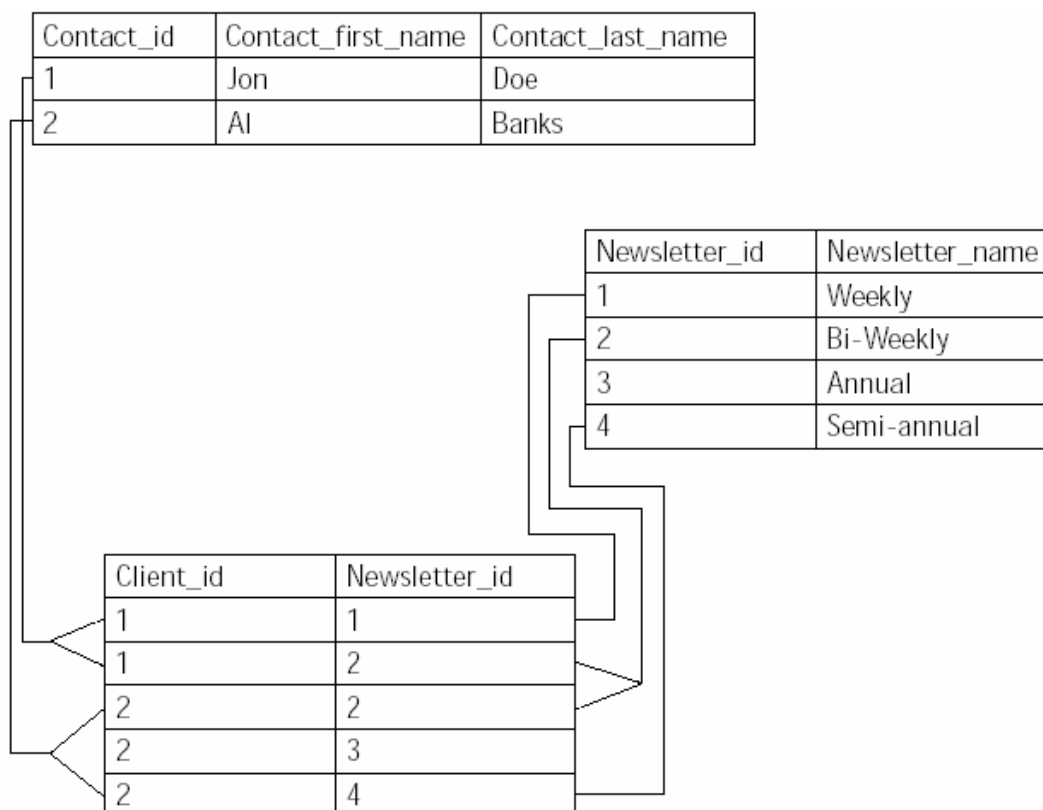
contact_id	contact_first_name	contact_last_name	newsletters
1	Jon	Doe	1,3,4
2	Al	Banks	2,3,4

Coloana *newsletters* conține mai mult de o valoare, fiind de fapt un șir de valori. Așa cum s-a menționat la *prima formă normală* această situație nu trebuie să apară niciodată în baza de date. Soluția este crearea încă unei tabele pentru a memora aceste informații, așa cum se exemplifică în tabelul 10.

Tabelul 10. Clienți ai publicațiilor

Client_id	Newsletter_id
1	1
1	2
2	2
2	3
2	4

Iată acum că această tabelă constituie baza unei relații m la n în baza de date:



8. Alte aspecte ale stocării datelor în BD relaționale

Integritatea referențială. Exemplele discutate până acum au folosit *chei străine*. O cheie străină este o coloană care referă o cheie primară dintr-o altă tabelă a bazei de date cu ajutorul căreia se realizează relația între cele două tabele. De exemplu, tabelele în 3 și 4:

company_id	company_name	company_address
1	BigCo Company	1121 43 rd St
2	LittleCo Company	4444 44 th St

contact_id	company_id	contact_name	contact_title	phone	email
1	1	Jay Greenspan	Vice President	4155551212	jgreen_1@yahoo.com
2	1	Brad Bulger	President	4155552222	bbulger@yahoo.com
3	2	John Doe	Lackey	2125556666	nobody@hotmail.com

Crearea și exploatarea bazelor de date relaționale

coloana *company_id* din tabela de contacte este o *cheie străină* către tabela de companii.

În anumite SGBD-uri, ca VFP, Oracle, Sybase sau PostGres, tabelele pot fi create cu definirea explicită a cheilor străine (*field validation/rule* în VFP) așa încât adăugarea va fi acceptată de către sistem doar dacă valoarea *cheii străine* există în tabela referită ca *cheie primară*.

În alte SGBD-uri, ca MySQL, cheile străine nu au această semnificație. În acest caz, programatorul trebuie să efectueze câțiva pași suplimentari înainte de a adăuga sau modifica înregistrări, cum ar fi (pentru tabelele 3 și 4):

- preia toate valorile pentru *company_id* din tabela de companii;
- verifică dacă valoarea pentru *company_id* pe care intenționezi să o inserezi în tabela de contacte există în șirul de date obținut mai sus;
- dacă ea există, inserează valorile;

Tranzacții. În BD relaționale au loc schimbări de apartenențe în grupuri. Multe schimbări necesită ca liniile să fie actualizate în mai multe tabele deodată. În unele cazuri acest lucru se face printr-o succesiune de instrucțiuni care preiau datele acolo unde trebuie să fie stocate.

Un site pentru comerțul electronic poate conține un cod care să efectueze operațiile:

1. adaugă clientul în tabela de clienți;
2. adaugă lista de cumpărături în tabela de cumpărături;
3. scade cantitățile comandate din tabela de stocuri.

Când se lucrează cu o serie de pași ca aceasta, există un potențial pentru probleme care pot apare. Dacă sistemul se blochează sau iese în decor între pașii 2 și 3, atunci baza de date conține date eronate.

Pentru a preveni o astfel de situație, unele SGBD-uri folosesc conceptul de *tranzacții*. Cu acesta, programatorul poate identifica un grup de comenzi. Dacă una dintre aceste comenzi eșuează în înregistrare, întregul grup este respins și baza de date este restaurată la starea sa înainte de efectuarea grupului de comenzi (tehnologia COMMIT/ROLLBACK).

Proceduri de memorare. SGBD-urile cu mai mare flexibilitate permit inserarea de cod procedural în baza de date (ceva foarte asemănător cu PHP și Perl). Sunt câteva avantaje care acest fapt le conferă:

- se poate reduce cantitatea de cod necesar pentru aplicațiile de mărime medie;
- se asigură baza de cunoștințe (proceduri) pentru execuția interogărilor și tranzacțiilor de pe sisteme de operare diferite și din SGBD-uri diferite în ceea ce privește sintaxa.

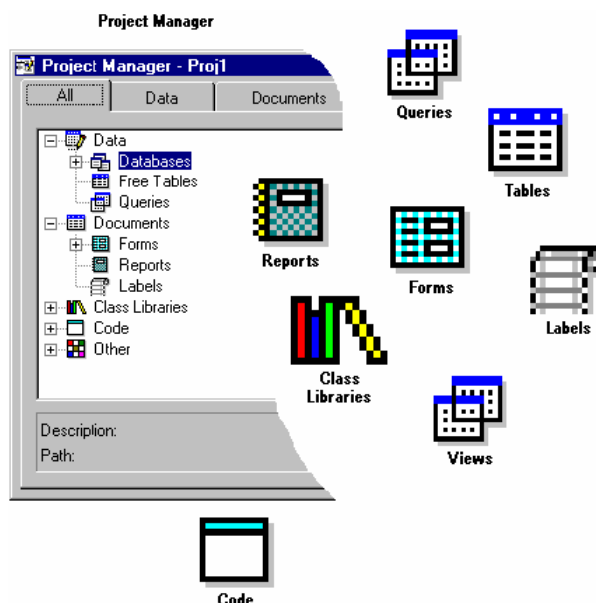
9. Lucrul cu Project Manager în VFP

Project Manager este instrumentul de organizare primară al lucrului cu date și obiecte în VFP. Un *proiect* este o colecție de fișiere, date, documente și obiecte VFP care sunt salvate ca un fișier cu extensia *.PJX*.

Se poate utiliza *Project Manager* pentru a organiza și manipula fișiere, crea tabele și baze de date, scrie interogări, defini forme și rapoarte și construi aplicații. Pentru construcția de aplicații, un bun îndrumar este *Programmer's Guide* din *MSDN*.

Pentru a crea un nou proiect, se urmează succesiunea de pași:

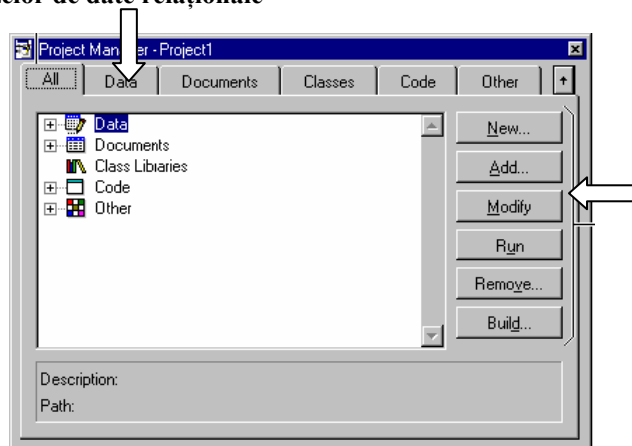
1. Se acționează butonul *New* din bara de instrumente;
2. Se selectează *Project* din caseta de butoane radio;
3. Se apasă butonul *New File* când se va activa o fereastră de dialog;
4. În caseta de editare cu eticheta *Enter project* se introduce numele dorit pentru proiect;
5. Se apasă butonul *Save* când se va genera un nou proiect cu numele ales.



Se pot acum adăuga proiectului tabele (fișiere *.DBF*) sau baze de date (fișiere *.DBC*).

În *Project Manager* datele sunt grupate pe categorii și prezentate într-o formă ierarhică. Pentru a urmări un tip de fișier sau obiect se activează boxa ☐ a grupului corespunzător. Activând tabulaturul *Data* putem restrânge domeniul de vizualizare al componentelor proiectului la nivelul de date, și anume: bazele de date, tabelele, interogările și vizualizările. Acționând unul din butoanele *New...*, *Add...*, *Modify*, *Run*, *Remove...* sau *Build...* se alege operația specifică asupra componentei sau categoriei selectate.

Acțiunile care se pot efectua asupra datelor sunt descrise în continuare.



10. Crearea tabelelor și indexurilor

Așa cum s-a arătat, pentru crearea unei tabele se poate folosi *Table Wizard* sau se poate lucra direct cu *Table Designer*.

Tipurile de date permise sunt descrise în tabelul următor.

Tabelul 11. Tipuri de date în VFP

Tip dată	Descriere	Exemplu
Character	Text alfanumeric	Adresa unui client
Currency	Unități monetare	Preț de cumpărare
Numeric	Numere întregi sau cu zecimele	Numărul de produse comandate
Float	La fel cu Numeric	
Date	Lună, zi și an	Data la care a fost făcută comanda
DateTime	Lună, zi, an, oră, minut și secundă	Data, ora la care un angajat vine la serviciu
Double	Număr în dublă precizie	Date provenite din experimente ce necesită înalt grad de precizie
Integer	Valori numerice fără zecimale	Numărul de înregistrare al unei comenzi
Logical	<i>True</i> sau <i>False</i>	Dacă o comandă a fost sau nu făcută
Memo	Text alfanumeric de lungime necunoscută	Lista apelurilor telefonice efectuate
General	OLE	Foaie de calcul din Excel
Character (Binary)	La fel cu Character dar valorile nu sunt translatate atunci când se schimbă codul de pagină	Parolele unor utilizatori stocate într-o tabelă și folosite în diferite țări
Memo (Binary)	La fel cu Memo dar valorile nu sunt translatate cu schimbarea codului de pagină	Scriptul de logare al unui utilizator (în diferite țări)

Dacă se dorește ca câmpul să accepte introducerea de valori nule, se check-ează butonul de pe coloana *NULL* din *Table Designer*.

Pentru a adăuga înregistrări în tabelă din *Project Manager* se parcurg următorii pași:

1. În *Project Manager* se selectează numele tablei;
2. Se apasă butonul *Browse*;
3. Din meniu, se selectează *View/Append Mode*;
4. Se introduc valorile noi înregistrări în fereastra *Browse*;

Crearea și exploatarea bazelor de date relaționale

Dacă se dorește vizualizarea fiecărui câmp pe linie separată se selectează din meniu *View/Edit*, revenirea la starea anterioară se face tot din meniu *View/Browse*.

În formatul *View/Browse* fiecare linie reprezintă o *înregistrare* iar fiecare coloană reprezintă *câmpurile* înregistrării.

Se pot crea (după cum se poate vedea din *Project Manager*) două tipuri de tabele: tabelele încorporate într-o bază de date (*database table*) și tabelele libere (*free table*), care sunt independente de orice bază de date.

Deplasarea într-o tabelă. Utilizând barele de defilare orizontale și verticale ne putem deplasa și vizualiza câmpuri diferite și înregistrări diferite. Se pot folosi de asemenea săgețile și TAB-ul. Pentru a accesa o anumită înregistrare, pentru a ne deplasa de la o înregistrare la alta, pentru a ajunge la începutul sau sfârșitul tabelii se poate accesa din meniu *Table/Got o Record >* când se activează un submeniu din care se alege opțiunea dorită.

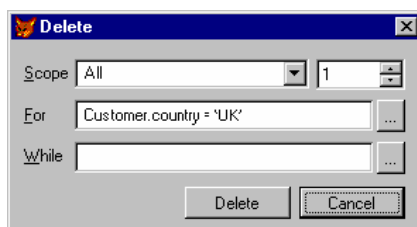
Pentru a edita câmpurile de tip Character, Numeric, Logical, Date sau DateTime este suficient să plasăm cursorul în celula dorită și să scriem valoarea dorită. Editarea câmpurilor Memo se face cu dublu click în câmpul dorit (sau CTRL+PgDn). Atunci se activează o fereastră de editare pentru conținutul câmpului memo. Un câmp general conține un obiect OLE (detalii la „<http://www.microsoft.com/data/oledb/prodinfo.htm>”) legat sau încapsulat. Se poate edita acest obiect cu dublu click în celula sa, când se va activa (sau se va încerca activarea) aplicației sale asociate pentru editare.

Ștergerea înregistrărilor. Ștergerea înregistrărilor dintr-o tabelă este un proces în 2 pași în VFP. Primul, marcarea înregistrărilor pentru ștergere de exemplu cu click pe boxa din stânga fiecărei înregistrări de șters. Al doilea, din meniu, *Table/Remove Deleted Records* care va șterge înregistrările marcate pentru ștergere. Aceasta este o ștergere permanentă și nu poate fi restaurată, spre deosebire de prima care este doar o ștergere logică și poate fi restaurată prin demarcarea înregistrărilor prin același procedeu. Eliminarea înregistrărilor șterse va închide tabela așa încât aceasta trebuie redeschisă pentru a putea fi utilizată.

Pentru a șterge mai multe înregistrări care verifică o condiție se poate folosi fereastra de dialog care se activează din meniu *Table/Delete Records...*

Analog, pentru a restaura înregistrări care verifică o condiție se poate accesa din meniu *Table/Recall Records...*

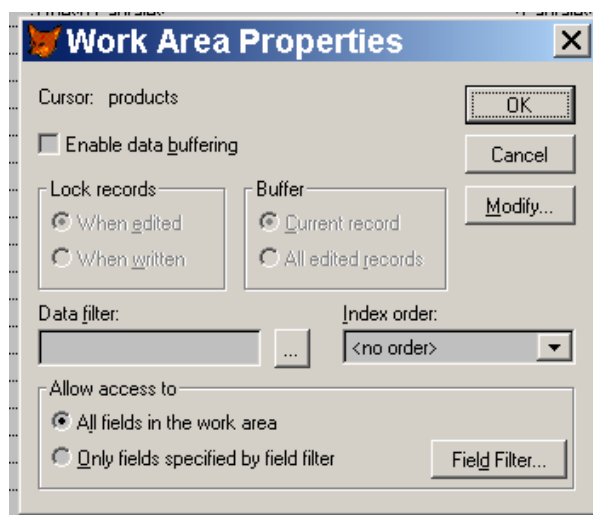
De exemplu, ca în figură, se pot selecta toate înregistrările care au valoarea UK în câmpul Country, folosind expresia **FOR Country = 'UK'**:



Fereastra de Browse se poate configura după dorință, prin (re)dimensionarea lățimii coloanelor, activarea sau dezactivarea liniilor de grid sau divizarea ferestrei de Browse în două porțiuni. Aceste operațiuni nu vor afecta structura actuală a tabelului.

Pentru a modifica structura unei tabeli, se alege în *Project Manager* opțiunea *Modify*. Structura tabelului este încărcată atunci în *Table Designer* și utilizatorul are posibilitatea să efectueze modificările dorite. Pentru a șterge un câmp din tabelă se selectează câmpul și se apasă butonul *Delete*.

Afișarea înregistrărilor în fereastra de Browse se poate face selectiv după o condiție impusă înregistrărilor. De asemenea se poate limita accesul la anumite câmpuri ale tabelului prin setarea unui filtru de câmpuri. Pentru impunerea unui filtru, din meniu, *Table/Properties*:



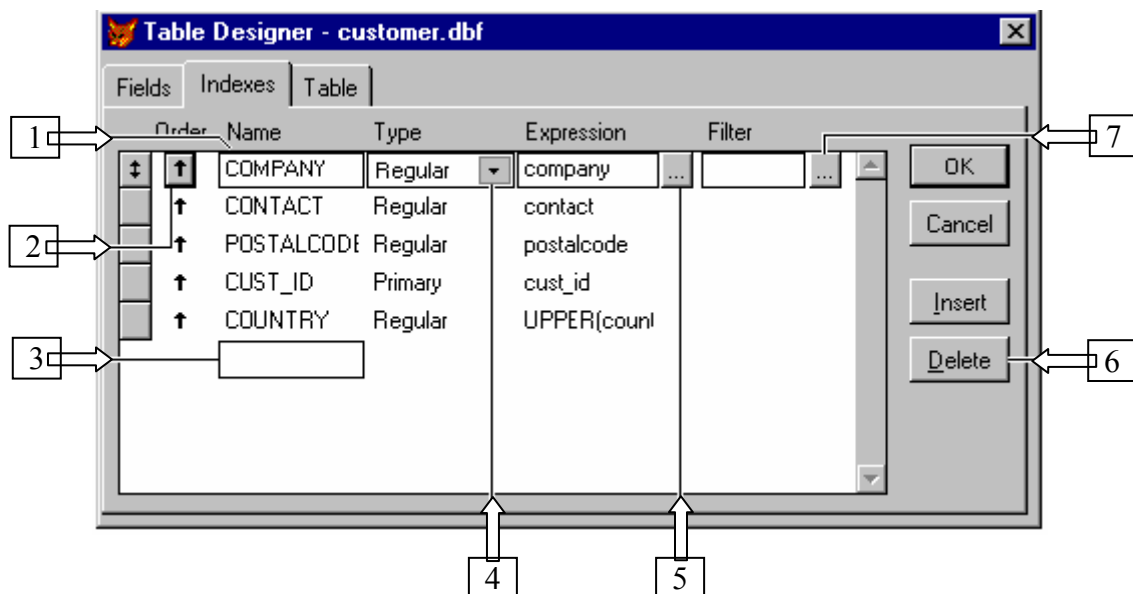
Odată ce a fost creată o tabelă, aceasta se poate ordona pentru a accelera regăsirea informației prin folosirea **indexurilor**. Cu indexurile, se pot procesa rapid înregistrările pentru afișare, interogare sau tipărire. Se pot selecta înregistrări, urmări dacă există valori duplicate într-un câmp și suportă stabilirea de relații între tabele în cadrul bazelor de date. Un index în VFP este o listă de numere de înregistrări care pointează către înregistrările corespunzătoare și astfel determină ordinea de procesare a înregistrărilor. Un index nu modifică ordinea în care înregistrările sunt stocate în tabelă, ci modifică doar ordinea în care acestea sunt citite de VFP.

Se poate crea mai mult de un index pentru o tabelă (fiecare index reprezentând câte o ordine de procesare a înregistrărilor din tabelă). Indexurile care se creează sunt stocate într-un fișier index cu structură compusă care este deschis și actualizat oricând tabela este folosită.

Crearea și exploatarea bazelor de date relaționale

Numele fișierului index este identic cu numele tabelului iar extensia sa este *.CDX*. Un index se poate crea ușor, așa încât frecvent se definește câte un index după fiecare câmp al tabelului. Un index se poate crea după un *câmp* sau după o *expresie*.

Din *Project Manager* se selectează tabela, se apasă butonul *Modify*, apoi în *Table Designer* se selectează tabulatorul *Indexes* când se activează o fereastră de forma:



în care avem posibilitatea:

- 1 să modificăm numele pentru un index;
- 4 să alegem tipul indexului după cum urmează:
 - *index primar* (cheie primară) unde doar valori unice sunt permise în câmp și determină ordinea de procesare a înregistrărilor; se pot crea câte un index primar pentru fiecare tabelă dintr-o bază de date; dacă tabela are deja un index primar, se poate crea atunci un *index candidat*;
 - *index candidat* care de asemenea necesită valori unice și determină ordinea de procesare a înregistrărilor; pot exista mai multe indexuri candidate pentru 1 tabelă;
 - *index regular* care permite duplicarea valorilor ce intră într-un câmp, este folosit pentru a determina ordinea de procesare și poate exista mai mult de 1 index regular pentru o tabelă;
 - *index unic* păstrat pentru compatibilitate cu versiunile mai vechi care selectează și ordonează un subset de înregistrări bazat pe prima apariție a unei valori într-un câmp specificat;
- 5 să definim o expresie simplă (formată din numele unui câmp) sau compusă (o expresie în care intervin nume de câmpuri din tabelă) după cum urmează:
 - câmpurile sunt evaluate în ordinea în care apar în expresie;
 - operatorul „+” aplicat la câmpuri numerice va aduna valorile din câmpuri;

employ.salary + employ.prime

- operatorul „+” aplicat la câmpuri caracter va concatena șirurile de caractere din câmpuri:

customer.country + customer.postalcode + customer.company

- sunt permise conversiile la tipul șir de caractere prin intermediul funcției STR(·):

STR(customer.maxordamt,20,4) + customer.company

- 7 să filtrăm înregistrările supuse indexării după o condiție logică:

employ.prime > 0 sau *customer.country = "Canada"*

- 2 să alegem o ordonare descendentă (↓) sau ascendentă (↑);
- 3 să adăugăm un nou index;
- 6 să ștergem un index;

Indexurile pot îndeplini diferite roluri, în funcție de tipul acestora:

Dacă vrei să	Folosește
Ordonezi înregistrările pentru a mări viteza de afișare, interogare sau tipărire	Un index regulat, primar sau candidat
Controlul intrării valorilor duplicat într-un câmp și ordonarea înregistrărilor	Un index primar sau candidat pentru tabelele libere

Odată creat, un index poate fi folosit la deschiderea unei tabele, în forma:

USE customer ORDER Cust_Id

când la activarea ferestrei de Browse se vor afișa înregistrările în ordinea specificată.

11. Colectarea tabelelor într-o bază de date

Punând tabelele într-o bază de date, se poate reduce stocarea datelor redundante și se poate proteja *integritatea* datelor.

Un SGBD cum este VFP furnizează mediul de lucru pentru:

- lucrul cu tabele;
- stabilirea relațiilor între tabele;
- setarea proprietăților și regulilor de validare pentru date.

Următorul exemplu arată cum se realizează o bază de date.

Problema:

Să presupunem că se dorește realizarea unei baze de date care să conțină informații despre universități și persoane de contact în cadrul acestora.

Așa cum s-a discutat până acum aceasta presupune crearea a cel puțin două tabele în cadrul bazei de date între care să se stabilească relații.

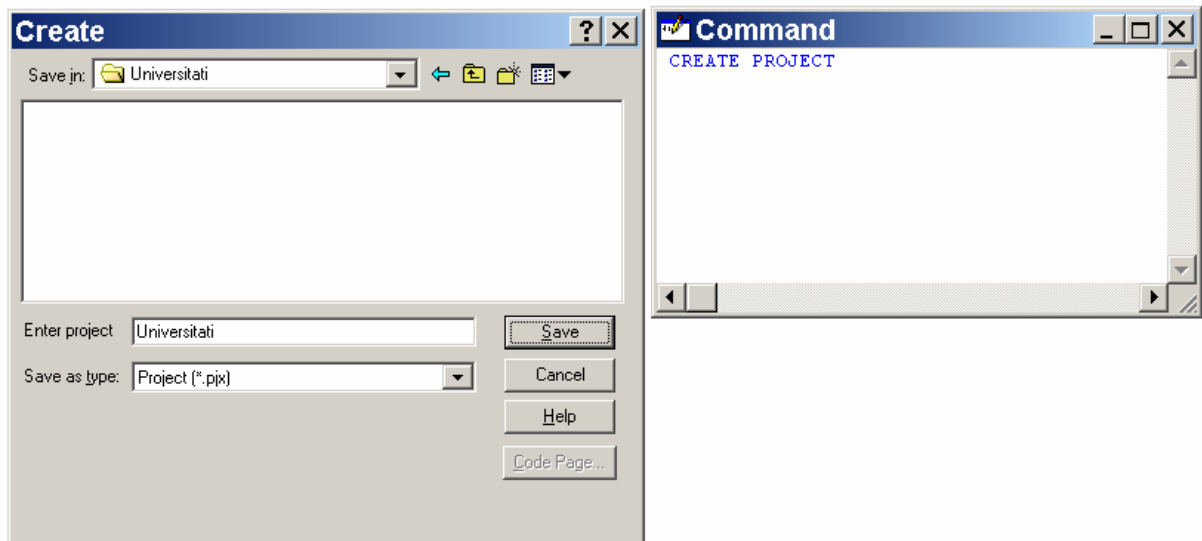
Crearea și exploatarea bazelor de date relaționale

Obiective:

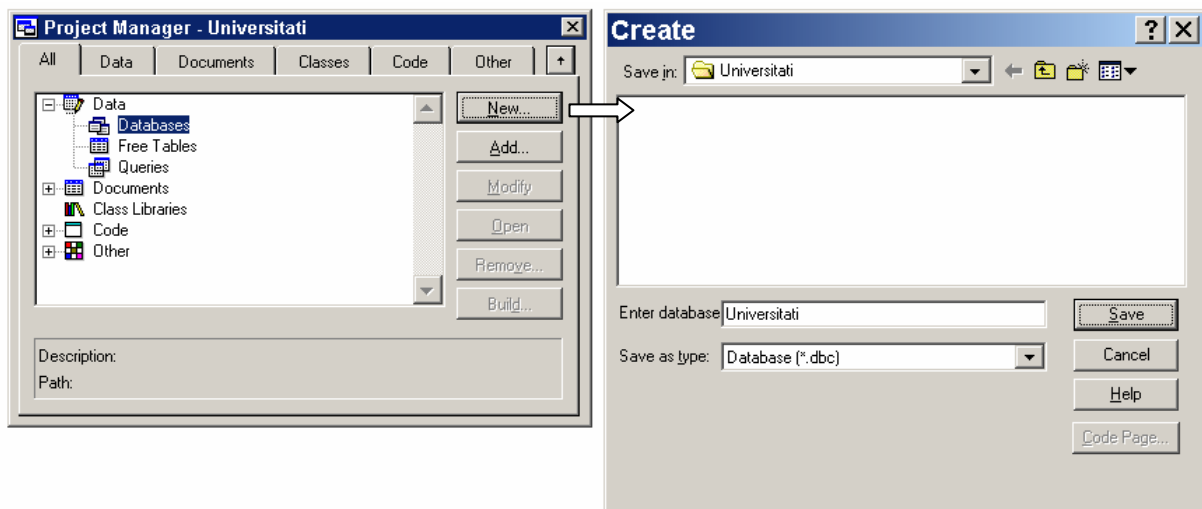
- crearea unui proiect (*Project Manager*);
- crearea unei baze de date (*Database Designer*);

Implementare:

- crearea proiectului
1. Din meniu sau din bara de instrumente se selectează *File/New (File Type: Project)/New file*;
 2. Se explorează calculatorul și se alege locul unde se dorește stocarea proiectului pe disc;
 3. Se creează un director pentru stocarea componentelor proiectului;
 4. Se dă nume proiectului; fie de exemplu numele Universitatii; acesta se va salva pe disc sub numele *Universitati.pjx*;



- crearea bazei de date



5. Din *Project Manager* din categoria *Data* se selectează *Databases* și se apasă butonul *New* și apoi *New Database*;

Crearea și exploatarea bazelor de date relaționale

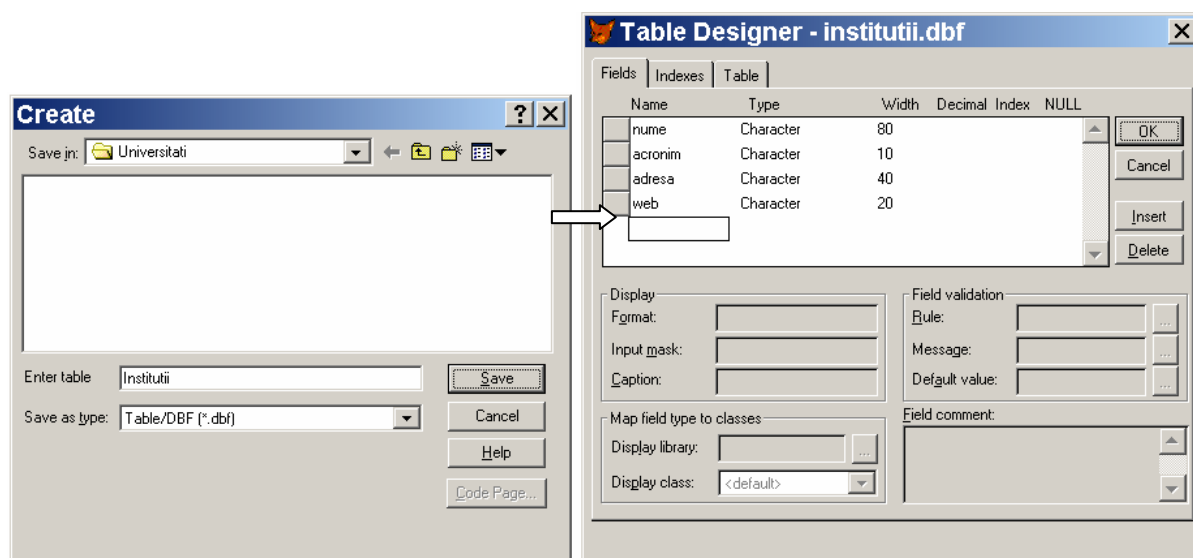
6. Se introduce un nume pentru noua bază de date din cadrul proiectului; fie aceasta *Universitati*; aceasta se va salva pe disc sub numele *Universitati.dbc*; se va încărca în mod automat aplicația expert *Database Designer*; cu ajutorul ei se construiesc tabelele bazei de date și se definesc relațiile între tabele; *Database Designer* mai permite crearea de vederi locale și de la distanță, editarea procedurilor stocate în baza de date (baza de cunoștințe pentru execuția interogărilor și tranzacțiilor de pe sisteme de operare diferite și din SGBD-uri diferite), realizarea de conectări la distanță;



- crearea tabelelor

Se enumeră informațiile care se doresc memorate în baza de date. Acestea ar pute fi: nume universitate, acronim, adresă, nume rector, prorectori, adrese email; se identifică faptul că pentru o universitate avem de memorat informații proprii instituției (nume, acronim, adresa, pagina web) și informații despre persoanele aflate la conducerea acesteia (nume, funcție, adresa email). Se desprind astfel în mod natural două tabele în baza de date: tabela institutii și tabela contacte.

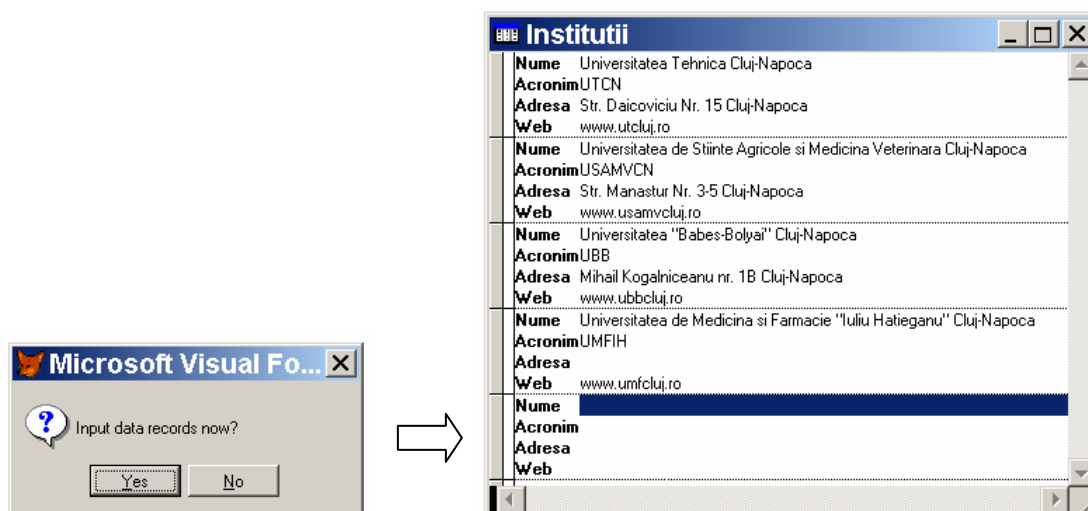
7. Din aplicația expert *Database Designer* se selectează butonul *New Table* și apoi din nou *New Table*; se va lansa în execuție în mod automat aplicația expert *Table Designer*;
8. Cu ajutorul aplicației *Table Designer* se creează tabela *institutii* cu structura *nume char(80)*, *acronim char(10)*, *adresa char(40)*, *web char (20)*; aceasta se va salva pe disc cu numele *Institutii.dbf*;



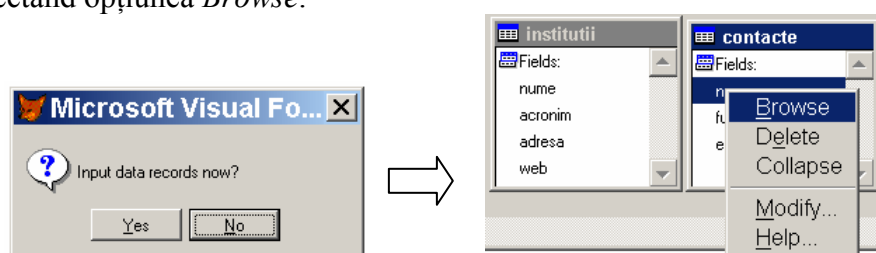
9. Se apasă butonul *Ok* când se activează o fereastră de dialog; aici se poate alege dacă să se introducă acum informațiile despre instituții sau mai târziu; să alegem introducerea acum;

Crearea și exploatarea bazelor de date relaționale

10. Se activează o fereastră care permite introducerea informațiilor; se introduc informațiile despre universități;



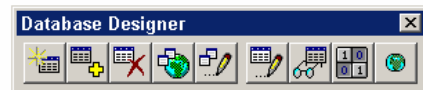
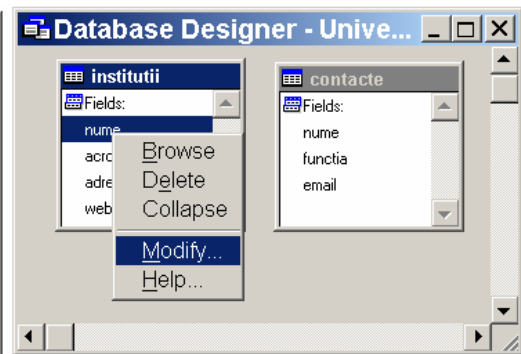
11. Din *Database Designer* se încarcă din nou *Table Designer* pentru crearea celei de-a doua table;
12. Se creează tabela *Contacte* cu structura: *nume char(25)*, *functia char(25)*, *email char(32)*;
13. Se alege să se introducă informațiile despre contacte mai târziu;
14. Se pot acum introduce informații în tabela *Contacte* activând click dreapta pe tabelă și selectând opțiunea *Browse*:



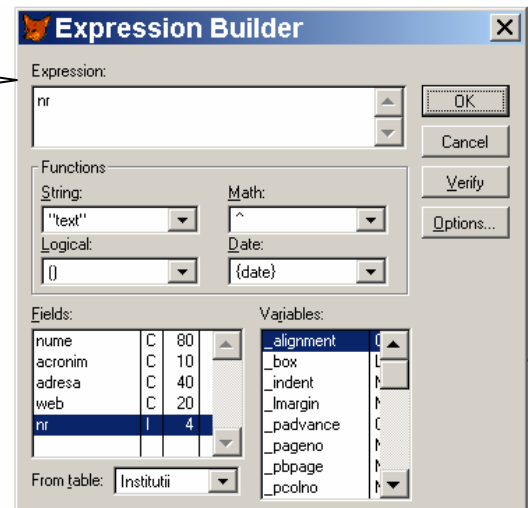
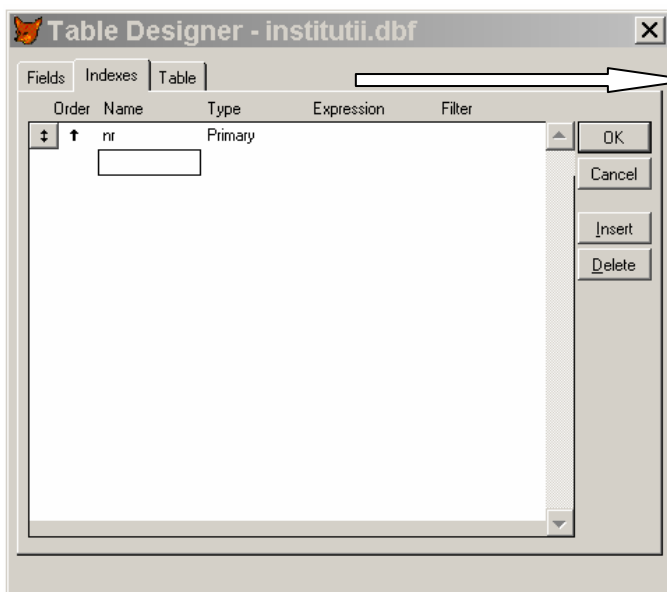
15. Se activează o fereastră *Browse*; se adaugă câte o înregistrare (din meniu, *Table/Append New Record* sau de la tastatură *Ctrl+Y*);
16. Se dorește stabilirea de relații între instituții și contacte; pentru aceasta este necesară adăugarea unor câmpuri numerice în table; deoarece la o instituție avem mai multe persoane de contact relația între table este de tipul 1 la n; pentru stabilirea relației este necesar ca valorile câmpului numeric din tabela *institutii* (fie acesta *nr int*) să fie distincte; se modifică tabela și se creează acest câmp în consecință;

Crearea și exploatarea bazelor de date relaționale

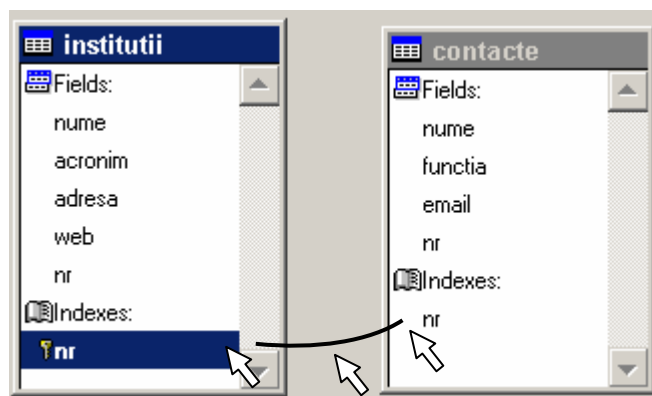
Contacte			
	Nume	Funcția	Email
	Gheorghe Lazea	Rector	Gheorghe.Lazea@aut.utcluj.ro
	Vasile Iancu	Prorector	Vasile.Iancu@mae.utcluj.ro
	Petru Berce	Prorector	Petru.Berce@cm.utcluj.ro
	Mihai Ilescu	Prorector	Mihai.Ilescu@cfdp.utcluj.ro
	Traian Canta	Secretar Stiintific	Traian.Canta@defplast.utcluj.ro
	Liviu Alexandru Marghitas	Rector	rector@hotmail.com
	Ioan Groza	Prorector	igroza@usamvcluj.ro
	Mihai Rusu	Prorector	mirusu@usamvcluj.ro
	Doru Pamfil	Prorector	dpamfil@usamvcluj.ro
	Andrei Marga	Rector	amarga@staff.ubbcluj.ro
	Nicolae Bocsan	Prorector	nbocsan@staff.ubbcluj.ro
	Nicolae Paina	Prorector	npaina@staff.ubbcluj.ro
	Paul Serban Agachi	Prorector	agachi@staff.ubbcluj.ro
	Vasile Cristea	Prorector	vcristea@staff.ubbcluj.ro
	Zoltan Kasa	Prorector	kasa@cs.ubbcluj.ro
	Wolfgang Breckner	Prorector	breckner@staff.ubbcluj.ro
	Arpad Neda	Prorector	neda@staff.ubbcluj.ro
	Mircea Miclae	Cancelar general	mmiclae@staff.ubbcluj.ro
	Marius Bojita	Rector	
	Mircea Grigorescu	Pro-Rector	
	Vasile Surcel	Pro-Rector	
	Andrei Achimas	Pro-Rector	
	Dumitru Zdrenghia	Secretar Stiintific	



17. Se adaugă un câmp numeric (de preferință cu același nume) în tabela *contacte* ale căror valori se vor completa ținând seama de apartenența persoanelor de contact la instituții;
- crearea indexurilor



18. Câmpul *nr* din tabela *institutii* se numește cheie primară; pentru stabilirea unei relații 1 la n după acest câmp tabela va trebui să fie indexată după acest câmp cu un *index primar*; din nou aplicăm *Modify* la tabela *Institutii* și îi asociem un index primar cu același nume cu câmpul după care se face indexarea: *nr*; se folosește aici aplicația expert *Expression Builder*; se va memora în mod automat pe disc indexul sub numele *Institutii.cdx*;
19. Câmpul *nr* din tabela *contacte* se numește cheie străină; valorile din această coloană a tabelului *contacte* nu sunt toate diferite între ele; după acest câmp tabela se poate indexa cu un index regulat; se creează indexul; fie numele acestuia *nr*;
- stabilirea relației între tabele

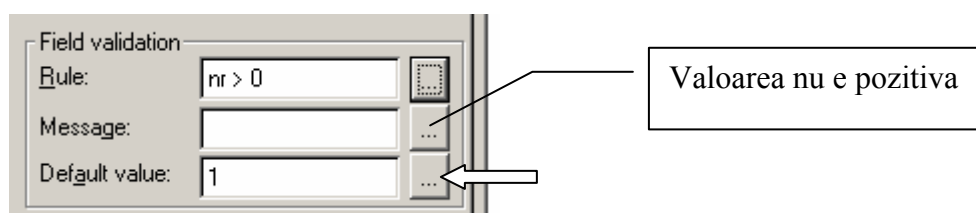


20. Se selectează indexul primar și se efectuează *drag and drop* cu mouse-ul peste indexul regulat (cheia primară peste cheia străină);

12. Validarea datelor la adăugare sau modificare

Pentru a controla tipul informației introduse de utilizator într-un *câmp* dintr-o tabelă se poate valida data independent de orice altă intrare în înregistrare, se poate realiza o **validare la nivel de câmp** a datelor. De exemplu, pentru a ne asigura că utilizatorul nu introduce o valoare negativă într-un câmp care trebuie să conțină doar valori pozitive.

Fie cazul când dorim să validăm printr-o regulă de acest tip câmpul *nr* din tabela *Contacte* a bazei de date *Universitati*. O soluție ar arăta astfel:

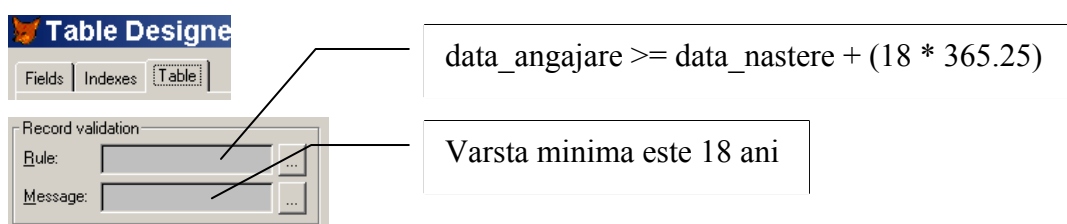


Butoanele „...” lansează aplicația expert *Expression Builder* cu ajutorul căreia se poate construi expresia pentru regula de validare a datelor, valoarea implicită a câmpului la adăugarea unei noi înregistrări și mesajul care se va afișa în caz de introducere eronată.

O altă posibilitate este de a stabili reguli de **validare la nivel de înregistrare** într-o tabelă. Presupunând că într-o tabelă *angajati* avem două câmpuri *data_nastere* și *data_angajare* de tip *date* atunci se poate stabili următoarea regulă de validare la nivel de înregistrare:

$$data_angajare \geq data_nastere + (18 * 365.25)$$

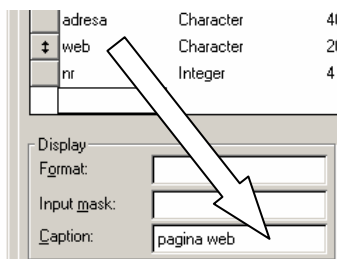
care să verifice că la angajare viitorul angajat are peste 18 ani:



Crearea și exploatarea bazelor de date relaționale

Incluzând comenzi sau funcții în regulile de validare care încearcă să mute pointerul de înregistrare în zona de lucru se pot provoca erori în stabilirea condițiilor.

Se poate defini un nume de câmp care va fi afișat (diferit de numele acestuia din tabelă) prin definirea acestuia în caseta de editare *Caption* din tabulaturul *Fields* al lui *Table Designer*:

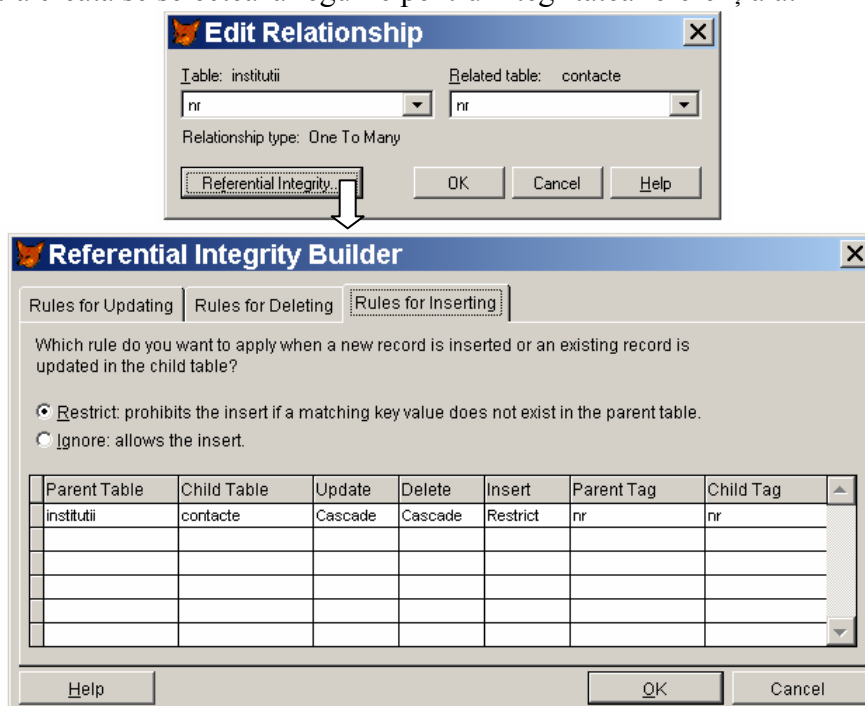


13. Manipularea înregistrărilor în baza de date și integritatea referențială

După ce au fost stabilite relațiile, se pot de asemenea defini reguli pentru manipularea înregistrărilor în baza de date. Acestea asigură *integritatea referențială*.

De exemplu, dacă se adaugă o companie în tabela *companii*, se dorește adăugarea automată a informațiilor despre persoana de contact în tabela *persoane_contact*. Pentru aceasta se folosește *Referential Integrity Builder*. Se urmează pașii:

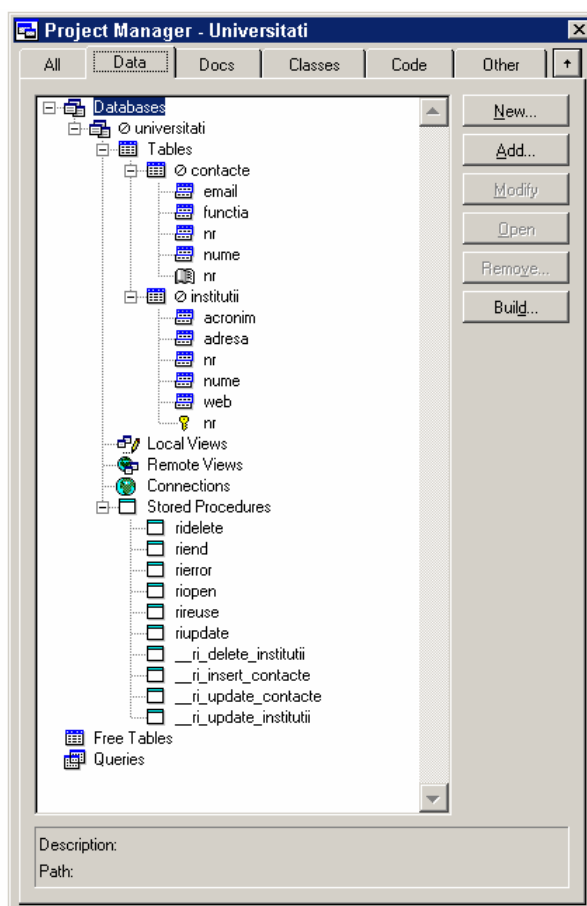
- cu baza de date deschisă din meniu se selectează *Database/Clean Up Database*;
- cu dublu click pe linia ce marchează relația se activează fereastra *Edit Relationship*;
- se apasă butonul *Referential Integrity...*;
- în fereastra creată se selectează regulile pentru integritatea referențială:



Referential Integrity Builder permite definirea următoarelor reguli:

Updating	Cascade	Atunci când o valoare a cheii primare este modificată actualizează toate înregistrările din tabela cu cheia străină corespunzătoare
	Restrict	Nu lasă modificarea valorii cheii primare atunci când aceasta are corespondente valori ale cheii străine în tabela cu cheia străină
	Ignore	Permite modificările în tabela cu cheia primară și lasă nereferite înregistrările din tabela cu cheia străină
Deleting	Cascade	Atunci când este ștearsă o înregistrare din tabela cu cheia primară șterge toate înregistrările din tabela cu cheia străină a căror valoare a cheii străine corespunde cu valoarea respectivă a cheii primare
	Restrict	Nu permite ștergerea unei înregistrări din tabela cu cheia primară atunci când valoarea cheii acesteia este regăsită printre valorile cheii străine din tabela cu cheia străină;
	Ignore	Permite ștergerile în tabela cu cheia primară și lasă nereferite înregistrările din tabela cu cheia străină
Inserting	Restrict	Atunci când o nouă înregistrare este adăugată sau una existentă este modificată în tabela cu cheia străină se verifică dacă există valoarea introdusă pentru cheia străină printre valorile cheii primare din tabela cu cheia primară și nu permite inserarea sau modificarea dacă această valoare nu este găsită
	Ignore	Permite adăugarea sau modificarea fără respectarea integrității referențiale

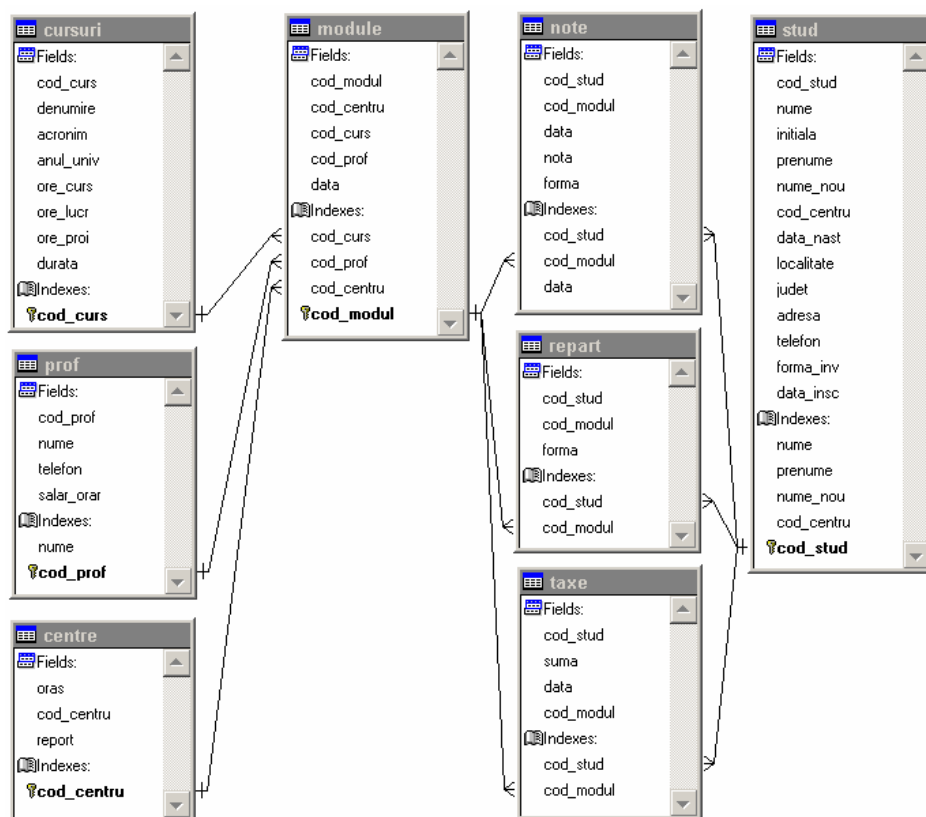
Referential Integrity Builder va genera cod de integritate referențială care va fi inclus în baza de date sub formă de proceduri, așa cum se vede din *Project Manager*.



Crearea și exploatarea bazelor de date relaționale

Aplicație:

Să se creeze baza de date cu cursanții Școlii Academice Postuniversitare de Informatică Aplicată și Programare.



Soluție:

Se creează o bază de date ce conține următoarele tabele (în care ↑ exprimă existența unui index după câmpul specificat; tipul indexului se va stabili în funcție de tipul relației tabelului respectiv cu celelalte în cadrul bazei de date).

Fișiere de date

9. Cursuri.dbf					2. Stud.dbf				
COD_CURS	N	5	↑		COD_STUD	N	5	↑↑	
DENUMIRE	C	70			NUME	C	20	↑↑	
ACRONIM	C	10			INITIALA	C	1		
ANUL_UNIV	C	12			PRENUME	C	30	↑↑	
ORE_CURS	N	2			NUME_NOU	C	20	↑↑	
ORE_LUCR	N	2			COD_CENTRU	N	5	↑↑	
ORE_PROI	N	2			DATA_NAST		D	8	
DURATA	N	2			LOCALITATE	C	20		
					JUDET	C	20		
					ADRESA	M	10		
					TELEFON		N	10	
					FORMA_INV		C	1	
					<i>S-scoala M-module</i>				

3. Repart.dbf					4. Taxe.dbf				
COD_STUD	N	5	↑		COD_STUD	N	5	↑	
COD_MODUL	N	5	↑		SUMA	N	10		
FORMA	C	1			DATA	D	8		
(Normal sau Distanță)					COD_MODUL	N	5	↑	
7. Module.dbf					8. Prof.dbf				
COD_MODUL	N	5	↑		COD_PROF	N	5	↑	
COD_CENTRU	N	5	↑		NUME	C	20	↑	
COD_CURS	N	5	↑		TELEFON	N	10		
COD_PROF	N	5	↑		SALAR_ORAR	N	10		
DATA	D	8							
12. Note.dbf					10. Centre.dbf				
COD_STUD	N	5	↑		ORAS	C	12	↑	
COD_MODUL	N	5	↑		COD_CENTRU	N	5	↑	
DATA	D				REPORT	N	12		
NOTA	N	2							

Rezultatul analizei problemei se prezintă în următorul tabel:

Nr	Nume	Index	Tip
12	note.dbf	Cod_stud	Regular
		Cod_modul	Regular
10	centre.dbf	Cod_centru	Primar
		Oras	Regular
9	cursuri.dbf	Cod_curs	Primar
8	prof.dbf	Cod_prof	Primar
		Nume	Regular
7	module.dbf	Cod_modul	Primar
		Cod_centru	Regular
		Cod_curs	Regular
		Cod_prof	Regular
		Natura	Regular
		Cod_prof	Regular
4	taxe.dbf	Localitate	Regular
3	repart.dbf	Cod_stud	Regular
		Cod_modul	Regular
2	stud.dbf	Cod_stud	Primar
		Nume	Regular
		Prenume	Regular
		Cod_centru	Regular

Analiza se face în modul următor (de exemplu pentru tabela 12): Tabela 12 (note.dbf) conține 2 indexuri: unul după cod student și unul după cod modul; valorile acestor câmpuri se pot repeta în tabelă: un student ia mai multe note (la diferite module) și un modul este frecventat de mai mulți studenți; indexurile sunt atunci regulate.

După ce s-au stabilit relațiile se impun:

- regulile de validare pentru câmpuri;

Crearea și exploatarea bazelor de date relaționale

- regulile de validare pentru înregistrări;
- regulile de integritate referențială.

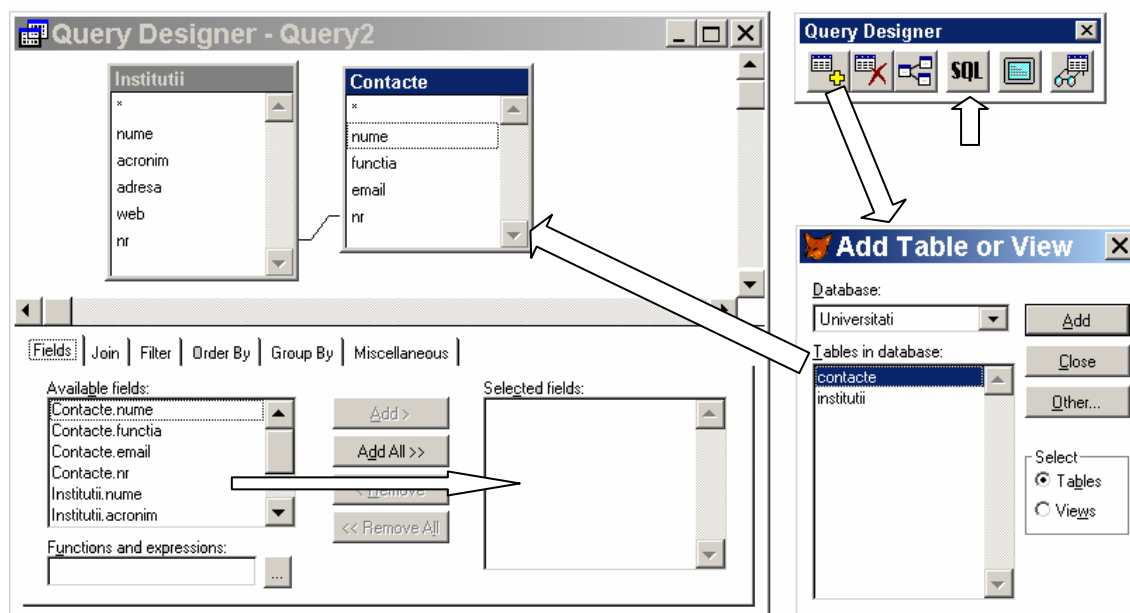
14. Interogarea unei baze de date și limbajul SQL

SQL este limbaj structurat pe interogări, bază de date interogativă și limbaj de programare. Utilizarea setului de *instrucțiuni SQL* este legată de crearea de interogări (*Query Designer*), definirea de vederi (*View Designer*), editarea de rapoarte (*Report Designer*). De asemenea, o comandă SQL poate fi dată în fereastra *Command* (*Window/Command Window*) sau inclusă în proceduri și programe.

Crearea de interogări și vizualizări

După ce a fost creată baza de date, au fost definite tabelele și relațiile între acestea, cum este cazul pentru baza de date *universitati* în care cheia primară pentru tabela *institutii* este *nr* iar tabela *contacte* are asociată cheia străină *nr* (care este un index regulat la tabelă), iar relația între cele două tabele este definită pe baza acestei relații, se poate crea o nouă interogare.

Pentru crearea unei noi interogări, din *Project Manager* se selectează *Queries*, se apasă butonul *New...* apoi *New Query*. Se lansează atunci aplicația expert *Query Designer*:



Practic, operațiile efectuate până în acest moment sunt rezultatul unor instrucțiuni simple în limbajul SQL, specificate prin intermediul wizard-ului pe care sistemul le-a executat pentru noi, instrucțiuni care se pot vizualiza de la butonul SQL:

```
FROM clienti!companii INNER JOIN clienti!persoane_contact ;  
ON Companii.company_id = Persoane_contact.company_id
```

Crearea și exploatarea bazelor de date relaționale

Se poate observa sintaxa acestor instrucțiuni:

```
FROM <nume_bd>.<nume_tabela_1> INNER JOIN <nume_bd>.<nume_tabela_2>;  
ON <nume_tabela_1>.<nume_câmp> = <nume_tabela_2>.<nume_câmp>
```

Se pot *selecta* acum câmpurile care să participe la interogare, ca în exemplul:



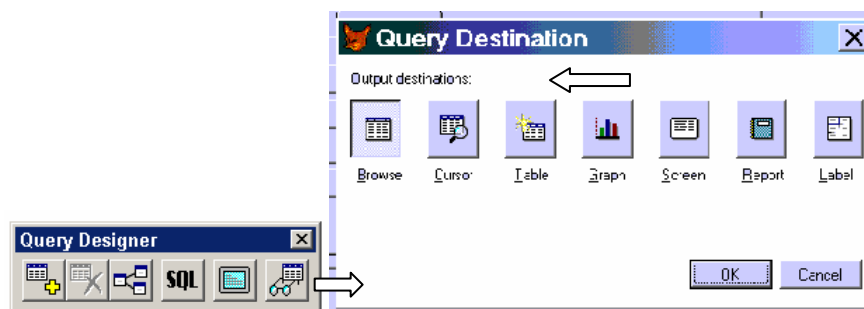
Din tabulatorul *Order By* se poate defini criteriul de ordonare în interogare. Să alegem *Institutii.nr* ca criteriu de ordonare. De asemenea, din tabulatorul *Group By* se poate defini un criteriu de grupare a rezultatului interogării. Să alegem ca criteriu de grupare *Institutii.nr*.

În acest caz interogarea va furniza ultima persoană de contact înregistrată pentru fiecare instituție. Dacă se renunță la grupare și se execută din nou interogarea se vor lista toate persoanele de contact împreună cu nume instituțiilor pe care le reprezintă.

Până aici, sistemul a completat pentru noi comenzile SQL corespunzătoare:

```
SELECT Contacte.nume, Contacte.functia, Institutii.nume, Institutii.nr,;  
Contacte.poz, Contacte.nr;  
FROM universitati!institutii INNER JOIN universitati!contacte ;  
ON Institutii.nr = Contacte.nr
```

Se poate acum specifica destinația interogării din *Query Designer*, când se activează o fereastră în forma:

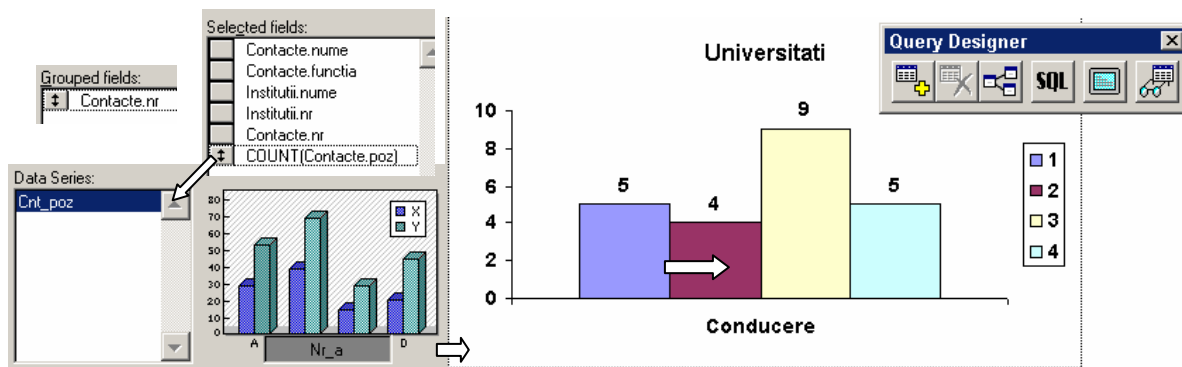


Între câmpuri numerice din tabele se pot crea grafice de corespondență (Butonul Graph). Astfel, pentru a face o reprezentare grafică din baza de date Universități să adăugăm câmpul *poz (int)* în tabela *Contacte*, și să atribuim câte o poziție fiecărei înregistrări.

Lansarea în execuție a interogării se poate face din meniu (*Query/Run Query*) sau din tastatură (*ctrl+Q*).

Astfel, dacă se selectează câmpurile *Institutii.nr* și *Count(Contacte.poz)* se poate obține cu ajutorul lui Graph Wizard un grafic de dependență a identificatorilor din tabele:

Crearea și exploatarea bazelor de date relaționale



În *Query Destination* se poate preciza ca destinație o fereastră de browse (*Butnul Browse*), caz în care rezultatul afișării este vizualizat într-o fereastră în care se pot parcurge înregistrările și câmpurile.

Opțiunea *Cursor* (*Butonul Cursor*) doar selectează înregistrările care corespund condițiilor specificate în interogare, fără a efectua afișări. Opțiunea *Table* (*Butonul Table*) permite salvarea rezultatului interogării într-o tabelă al cărui nume se va introduce la execuția interogării. Opțiunea *Screen* (*Butonul Screen*) permite afișarea rezultatului interogării la imprimantă sau trimiterea sa într-un fișier text. Opțiunile *Report* și *Label* (*Butoanele Report* și *Label*) permite trimiterea rezultatului interogării către un raport sau etichetă, a căror construcție se va discuta mai târziu.

Aceeași pași care au fost urmați în crearea unei interogări se efectuează și pentru crearea unei vederi.

Vederile combină calitățile tabelelor cu cele ale interogărilor; la fel ca în cazul interogărilor, se poate crea o vedere pentru extragerea unui set de date din una sau mai multe tabele asociate; la fel ca la o tabelă, o vedere poate fi folosită pentru actualizarea informațiilor și stocarea permanentă a informațiilor pe hard-disk. Vederile pot fi însă folosite și la culegerea și modificarea datelor off-line în afara sistemului principal.

Dacă se dorește accesul la date stocate pe un server la distanță, trebuie creată o vedere externă. În acest scop, trebuie întâi să ne conectăm la o sursă de date. O sursă de date externă este de obicei un server extern pentru care este instalat un driver ODBC (open database connectivity) și s-a atribuit un nume sursei de date ODBC. Pentru a beneficia de o sursă de date, trebuie să fie instalat driverul ODBC. Din mediul VFP se poate atunci defini sursa de date și conexiunile.

15. Crearea de vederi locale

O vedere locală permite să extragem înregistrări dintr-o tabelă, să efectuăm modificări asupra înregistrărilor extrase și să transmitem modificările în tabelele sursă.

Se pot crea vederi din tabele locale, din alte vederi, din tabele stocate pe un server, din surse de date situate la distanță, cum ar fi Serverul SQL Mirosoft prin intermediul protocolului ODBC. Se poate configura VFP să efectueze modificările în tabelele sursă în momentul efectuării modificărilor în vedere.

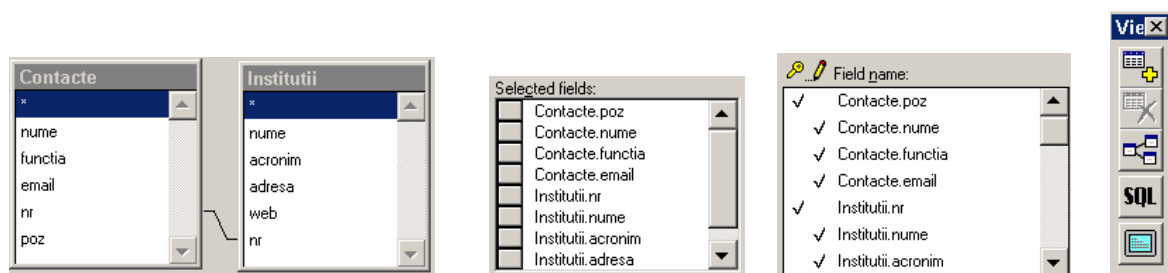
O vedere locală se poate crea cu ajutorul aplicației expert *View Designer*. Astfel, avem următoarele posibilități:

1. Din fereastra de comenzi (*Window/Command Window*) cu comanda *create sql view*;
2. Din bara de instrumente: *New/View/New file* sau din meniu *File/New/View/New file*;
3. Din *Project Manager* se selectează o bază de date, se alege *Local Views* și butonul *New*;

Se poate folosi *View Designer* pentru a crea vederi din baza de date *Universitati.dbc*. Se pot construi două vederi: una care să redea conținutul bazei de date ordonat alfabetic crescător după persoanele de contact și alta care să redea conținutul ordonat alfabetic crescător după numele universității și apoi alfabetic descrescător după funcție.

Vederea bazei de date Universitati după persoanele de contact

1. Se deschide proiectul *Universitati.pjx* (*Open/Project*);
2. Se selectează baza de date *Universitati*;
3. Se expandează conținutul bazei de date ($\oplus \rightarrow \boxminus$);
4. Se selectează din aceasta *Local views*;
5. Se apasă butonul *New...* și se alege opțiunea *New View*;
6. La fel ca la interogări, se includ tabelele *contacte* și *institutii* în *View Designer*;



7. Se selectează toate câmpurile, mai puțin *Contacte.nr*;
8. Se alege relația de ordine dorită (*Order By/Ordering criteria: Contacte.nume* ↑);
9. În tabulatorul *Update Criteria* se precizează cheile primare *Contacte.poz* și *Institutii.nr*;
10. Se apasă butonul *Update All* pentru a face posibilă modificarea tuturor câmpurilor din tabele pe baza modificărilor efectuate în vedere; comanda SQL transmisă sistemului se poate vedea din bara de instrumente a lui *View Designer*.

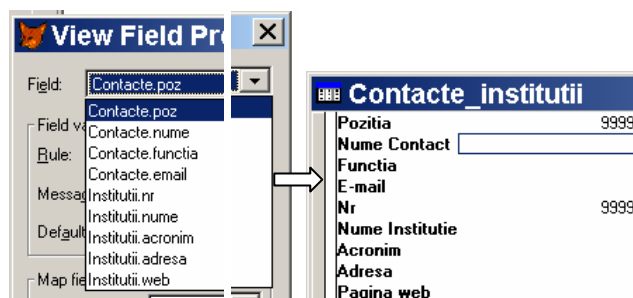
Crearea și exploatarea bazelor de date relaționale

```
SELECT Contacte.poz, Contacte.nume, Contacte.functia, Contacte.email,;  
Institutii.nr, Institutii.nume, Institutii.acronim, Institutii.adresa,;  
Institutii.web;  
FROM universitati!institutii INNER JOIN universitati!contacte ;  
ON Institutii.nr = Contacte.nr;  
ORDER BY Contacte.nume
```

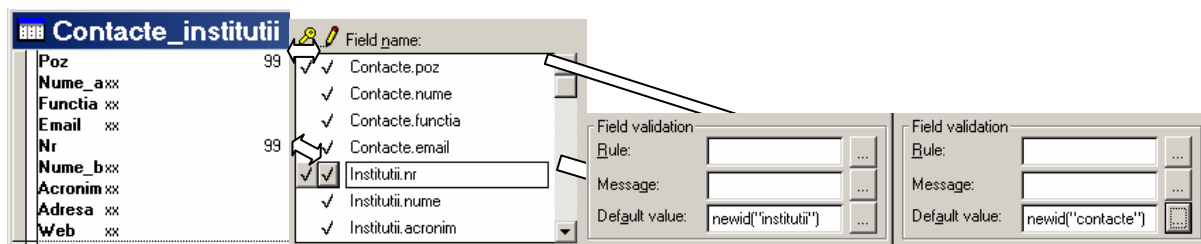
11. Se salvează vederea din bara de instrumente (*Save*) sau din meniu (*File/Save*) de exemplu cu numele *contacte_institutii*;
12. Se pot acum defini opțiuni de vizualizare ale câmpurilor vederii; se selectează vederea *contacte_institutii* din *Project Manager* și se apasă *Modify*;



13. În *View Designer* la tabulaturul *Fields* se apasă butonul *Properties*;
14. Se pot aici defini reguli de validare pentru câmpurile din vedere (*Field validation*) la fel ca la tabele și etichete pentru afișarea lor în fereastra de *Browse* (*Display/Caption*);
15. Se setează din această fereastră de dialog proprietățile pentru toate câmpurile vederii;



16. Se pot adăuga acum persoane de contact și instituțiile corespunzătoare dacă se lansează în execuție vederea creată: *Query/Run Query* când se poate afișa în forma *Append Mode* și *Edit* din meniu de la *View*;
17. Se poate seta opțiunea pentru a efectua modificările în tabelele incluse în vedere; pentru aceasta se selectează opțiunea *Send SQL updates* din tabulaturul *Update Criteria* al vederii *contacte_institutii* cu ajutorul lui *View Designer*;
18. Se pot acum adăuga persoane de contact și instituții, modificarea efectuându-se automat în tabelele bazei de date; pentru aceasta, se execută vederea (*Ctrl+Q*) și se adaugă o nouă înregistrare în aceasta (*Ctrl+Y*);
19. Pentru ca modificările să devină active trebuie închis proiectul (*Close all data* în fereastra de comenzi);
20. La o nouă deschidere a acestuia se pot observa modificările în tabele;



21. Pentru a insera noile chei pentru instituție și contact este necesară selectarea explicită a modificării în vedere (*View Designer/Update Criteria/Field name*);
22. Sistemul salvează vederea în baza de date sub forma unui tabel liber cu același nume cu vederea, care se poate observa în *Project Manager* la încărcarea aplicației expert *Database Designer*;

Vederea bazei de date Universitati după institutii și apoi persoane de contact

23. Se urmează acești succesiune de pași ca mai sus, însă se alege la relația de ordine dorită (*Order By/Ordering criteria: Instituti.nume ↑ și Contacte.nume ↓*);

Stocarea de proceduri în baza de date pentru câmpurile autoincrement

Baza de date *Universitati* este acum pregătită pentru a construi o metodă de autoincrementare a valorilor pentru cheile primare. Este necesară crearea în baza de date a unei tabele care să memoreze viitoarele valori ale cheilor primare pentru fiecare tabelă în parte. Ulterior, se folosește această tabelă de către o procedură stocată în baza de date pentru a stabili valorile de increment. Se urmează pașii:

1. Se adaugă în *Universitati.bdc* tabela *auto_id* cu structura (*id int; tabela char(50)*);
2. Se completează *Auto_id.dbf* cu valorile corespunzătoare;
3. Se indexează *Auto_id.dbf* după cele două câmpuri ale sale cu indecși regulari;
4. Se creează o funcție pentru autoincrementare; fie aceasta *NouId()*; Se poate selecta codul procedurii *NewId()* din baza de date *Books.dbc* din subdirectorul de instalare al *VFP: Wizards/Template/Books/Data* și copia în baza de date *Universitati.dbc*; Acesta se modifică corespunzător, ținând cont de numele actuale:

```

FUNCTION NouID(tcAlias)
  LOCAL lcAlias, ;
    lnID, ;
    lcOldReprocess, ;
    lnOldArea
  lnOldArea = SELECT()
  lnID = 0
  IF PARAMETERS() < 1
    lcAlias = UPPER(ALIAS())
  ELSE
    lcAlias = UPPER(tcAlias)
  ENDIF
  lcOldReprocess = SET('REPROCESS')
  *-- Lock until user presses Esc
  SET REPROCESS TO AUTOMATIC

```

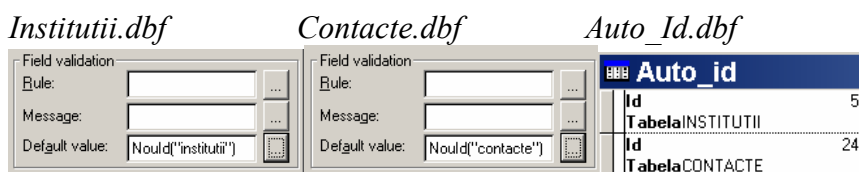
```

IF !USED("AUTO_ID")
  USE universitati!auto_id IN 0
ENDIF
SELECT auto_id
IF SEEK(lcAlias, "auto_id", "tabela")
  IF RLOCK()
    lnID = auto_id.id
    REPLACE auto_id.id WITH auto_id.id + 1
  UNLOCK
ENDIF
ENDIF
SELECT (lnOldArea)
SET REPROCESS TO lcOldReprocess
RETURN lnID
ENDFUNC

```

Crearea și exploatarea bazelor de date relaționale

5. Se exploatează funcția *Nould()* prin definirea autoincrementelor în tabelele *Institutii* și *Contacte*;

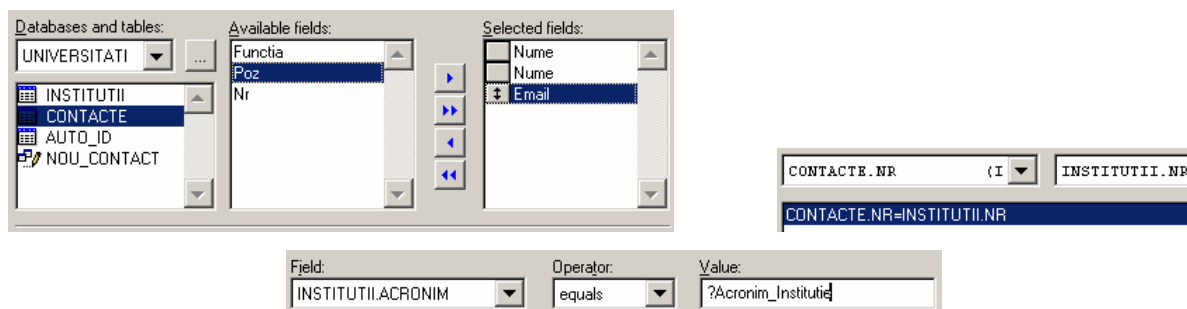


6. Se construiește vederea *Nou_Contact* pe baza tabelor *Contacte* și *Institutii*; se includ toate câmpurile din tabela *Contacte* și câmpul nume din tabela *Institutii* și se procedează identic cu cazul anterior;
7. Se construiește vederea *Noua_Institutie* pe baza tabelii *Institutie*, în care se includ toate câmpurile din tabelă; se procedează la fel cu cazurile anterioare;

Utilizarea Wizard-ului pentru crearea de vederi

Pentru vizualizarea tuturor persoanelor care aparțin unei instituții cu un anumit acronim se poate realiza o vedere parametrizată care să le includă. Se urmează pașii:

1. Se selectează *Local Views*; Se activează *New../View Wizard*;
2. Se aleg câmpurile *contacte.numa*, *institutii.numa*, *contacte.email*, *institutii.nr* și *contacte.nr*;



3. Se definește relația *institutii.nr = contacte.nr*;
4. Se includ toate liniile din tabela *Contacte* (*All Rows from this table*);
5. Se definește interogarea în tabela *Institutii* după *Acronim* în forma *?Acronim_Institutie*;
6. Se salvează vederea sub numele *CautContact*;
7. Se execută interogarea (*Run Query*);

În același mod se poate defini vederi care să extragă o persoană de contact pentru o anume universitate sau să găsească pagina web a unei universități.

16. Lucrul cu fereastra de comenzi

Pe lângă facilitățile oferite de mediul vizual, sistemul VFP pune la dispoziția utilizatorului și o interfață de tip text, în care utilizatorul are posibilitatea să lanseze comenzi sistemului. Există un set de comenzi pe care sistemul le acceptă și le execută din fereastra de comenzi.

Se pot lansa comenzi practic pentru orice proces care poate fi accesat din meniuri. Cele mai importante comenzi sunt redate în continuare.

Vizualizarea structurii tabeli curente se realizează cu comenzile:

```
LIST STRUCTURE [TO PRINTER [PROMPT] | TO FILE <file>]
```

```
DISPLAY STRUCTURE [IN <expN> | <expC>] [TO PRINTER [PROMPT] | TO FILE <file>]
```

unde:

IN <expN | expC> specifică numărul zonei de lucru sau aliasul tabeli,

TO PRINTER trimite informația la imprimantă,

PROMPT introduce dialog (confirmare) înainte de imprimare,

TO FILE <nume_fis> salvează informație în fișier,

Exemplu:

```
DISPLAY STRUCTURE IN 3 TO FILE <str_baz1.txt>
```

Aliasul tabeli se poate vizualiza din meniu la opțiunea *Window/Data Session*.

Structura unei baze se poate ulterior modifica în sensul adăugării/eliminării unor câmpuri sau modificarea mărimii câmpurilor prin comanda:

```
MODIFY STRUCTURE
```

Comenzile DISPLAY și LIST sunt folosite pentru afișarea conținutului tabelor.

```
DISPLAY [ [FIELDS] <lista campurilor> ]
```

```
[<domeniu>] [FOR <expL1>] [WHILE <expL2>] [OFF]
```

```
[TO PRINTER [PROMPT] | TO FILE <numefis>] [NOCONSOLE] [NOOPTIMIZE]
```

Exemplu:

```
USE Contacte
```

```
DISPLAY ALL
```

afișează toate câmpurile și înregistrările cu oprire la fiecare pagină. Pentru afișarea numai a câmpurilor specificate lansăm de exemplu comanda:

```
DISPLAY ALL FIELDS NUME, EMAIL
```

Afișarea înregistrărilor care îndeplinesc o condiție (au valoarea câmpului POZ mai mare decât 10):

```
DISPLAY ALL FOR POZ > 10
```

Afișarea înregistrărilor care au șirul de caractere din câmpul nume mai mare sau egal decât șirul Ha:

Crearea și exploatarea bazelor de date relaționale

DISPLAY ALL FOR NUME >= „Ha”

DISPLAY ALL WHILE NUME > „Ha”

afișează înregistrările cât timp expresia logică NUME > „Ha” este adevărată.

DISPLAY ALL OFF

nu se afișează coloana 0 ce conține numărul de ordine al înregistrărilor

DISPLAY

afișează numai înregistrarea curentă.

În cadrul expresiei logice pot fi folosiți operatorii logici AND, OR, NOT

DISPLAY NEXT 5 FOR nume < 'R' AND nume > 'A'

Comanda LIST are sintaxa:

LIST

[FIELDS <expr list>]

[<scope>] [FOR <expL1>] [WHILE<expL1>] [OFF]

[NOCONSOLE] [NOOPTIMIZE] [TO PRINTER [PROMPT] | TO FILE <file>]

LIST afișează toate înregistrările fiind echivalentă cu comanda DISPLAY ALL;

LIST NEXT 3 afișează următoarele 3 înregistrări și mută indicatorul corespunzător (pe a 3-a înregistrare);

Exemple:

LIST FOR NOT NUME= ' Giurgiu'

LIST FOR "Gi" \$ nume

afișează înregistrările ce conțin șirul “Gi” în câmpul nume.

LIST FOR LIKE ("*iurg*", nume)

Zona de lucru este o zonă de memorie rezervată pentru păstrarea informațiilor necesare lucrului cu o tabelă. Zona de lucru 1 este curentă implicit după lansarea FoxPro. La un moment dat o singură zonă este activă.

Prin comanda SELECT <expN> | <expC> se stabilește zona de lucru curentă:

SELECT 7

Zona de lucru indice 7 a devenit curentă, activată.

Funcția SELECT () returnează numărul zonei de lucru curente.

Operatorul ? este folosit pentru afișarea unei valori pe ecran. Astfel,

? SELECT()

va afișa:

7

Pentru a lucra cu o tabelă ea trebuie deschisă. Deschiderea tabelii înseamnă înscrierea caracteristicilor esențiale ale bazei (structura, numărul înregistrărilor, etc.) într-o anumită zonă de lucru, nu neapărat cea curentă.

Crearea și exploatarea bazelor de date relaționale

Funcția USED (expN | expC) returnează .T. dacă în zona de lucru specificată prin argument este deschisă o tabelă (în caz contrar returnează .F.). Argumentul expC al funcției se referă la aliasul tabelului (un alt nume al tabelului).

? USED(3)

.F.

Funcția DBF(expN | expC) returnează numele tabelului deschise în zona specificată prin argument sau numele bazei de alias expC:

SELECT 3

? SELECT() && se afișează 3

USE tabela1

LIST STRUCTURE

? USED(3) && se afișează .T.

? DBF(3) && se afișează tabela1

USE && se închide tabela1 din 3

? USED(3) && afișează .F.

USE tabela1 IN 4 && se deschide tabela1 în zona de lucru 4

LIST STRUCTURE && nu se listează structura

USE tabela1 IN 3 AGAIN && se deschide tabela *tabela1* în zona de lucru curentă 3 && rămânând deschisă și în zona 4

Close all data

Închide toate tabelurile din mediul de lucru și odată cu acestea toate bazele de date și proiectele ce le utilizează.

Poziționarea pe o înregistrare se poate face cu ajutorul comenzilor GO, GOTO și SKIP. FoxPro atribuie în mod reflex un număr de înregistrare fiecărui articol din fișierul deschis.

GO [RECORD] <expN1> [IN <expN2> | IN <expC>]

GO TOP | BOTTOM [IN <expN2> | IN <expC>]

GOTO [RECORD] <expN1> [IN <expN2> | IN <expC>]

GOTO TOP | BOTTOM [IN <expN2> | IN <expC>]

Comanda GO număr_articol permite trecerea imediată pe articolul având numărul specificat (expN1) :

Go 20

Disp

Este suficient să precizăm primele 4 litere din numele comenzii pentru ca sistemul să recunoască comanda.

Pentru saltul peste un anumit număr de înregistrări se poate folosi comanda SKIP

SKIP -1 && acceptă și valori negative, pentru saltul înapoi;

Crearea și exploatarea bazelor de date relaționale

Cu ajutorul comenzii SET se poate defini o listă de câmpuri implicită la afișare. De exemplu:

```
SET FIELDS TO NUME, EMAIL
```

```
LIST
```

```
SET FIELDS OFF
```

```
LIST
```

Filtrele pentru înregistrări se definesc tot cu ajutorul comenzii SET:

```
SET FILTER TO POZ>10
```

```
LIST
```

```
SET FILTER TO
```

```
LIST
```

Pentru modificarea înregistrărilor din fișier se pot folosi comenzile EDIT, BROWSE, REPLACE și CHANGE.

Comanda EDIT modifică înregistrarea curentă. Comanda EDIT 4 modifică înregistrarea 4.

```
EDIT FIELDS NUME, EMAIL (restricționare la câmpurile enumerate)
```

```
EDIT FOR nr = 4 (restricționare la înregistrările care satisfac condiția).
```

Comanda REPLACE permite un alt mod de modificare a valorii unor câmpuri. Comanda are sintaxa :

```
REPLACE <field1> WITH <expr1> [ADDITIVE] [, <field2> WITH <expr2> [ADDITIVE]] ... [<scope>]  
[FOR <expL1>] [WHILE <expL2>]
```

unde <scope>: ALL, NEXT <expN>, RECORD <expN>, sau REST

Exemple:

```
use Institutii
```

```
goto 3
```

```
REPLACE nume WITH "Popescu"
```

```
use Contacte
```

```
REPLACE ALL poz WITH poz*2
```

```
REPLACE ALL nume WITH UPPER(nume)
```

Opțiunea ADITIVE operează numai pentru câmpuri de tip memo. Dacă ea este folosită, valoarea indicată după WITH va fi adăugată la sfârșitul conținutului curent al câmpului.

Suprimarea înregistrărilor (se realizează în doi pași):

Comanda DELETE marchează înregistrările ca fiind șterse dar nu le șterge efectiv din fișier, și apoi cu comanda PACK se reorganizează fișierul suprimând efectiv articolele marcate (marcarea înregistrărilor se poate face și din fereastra comenzii BROWSE).

Comanda DELETE permite ștergerea unuia sau mai multor articole:

Crearea și exploatarea bazelor de date relaționale

DELETE [<scope>] [FOR <expL1>] [WHILE <expL2>]

<scope>: ALL, NEXT <expN>, RECORD <expN>, sau REST

Exemple :

DELETE RECORD 12 șterge articolul 12

DELETE NEXT 4 șterge următoarele 4 articole

DELETE ALL FOR nr=4

DELETE ALL șterge toate articolele

DELETE REST șterge articolele de la poziția curentă până la sfârșitul fișierului.

Comanda RECALL restabilește înregistrările ștergute cu DELETE.

RECALL [<scope>] [FOR <expL1>] [WHILE <expL2>]

<scope>: ALL, NEXT <expN>, RECORD <expN>, sau REST

Indexarea unui tabel se poate realiza din fereastra de comenzi.

INDEX ON câmp TAG câmp DESCENDING

Dacă se dorește o indexare după mai multe câmpuri este necesară o comandă de tipul:

INDEX ON câmp1+câmp2 TAG nume

De exemplu, dacă se dorește o indexare după nume și apoi, pentru același nume după cifra de afaceri, se poate folosi o indexare în forma:

INDEX ON nume+STR(ca) TAG x

În exemplu s-a convertit cifra de afaceri ca în șir caractere (cu ajutorul funcției STR) și s-au concatenat șirurile de caractere nume și STR(ca). La redeschiderea fișierului astfel indexat se va putea folosi de exemplu :

USE parteneri ORDER TAG x

Dacă în timpul exploatării se dorește trecerea la indexarea după un alt index se folosește comanda SET ORDER TO:

USE contacte ORDER nume

LIST

SET ORDER TO poz

LIST

Într-un fișier indexat se pot efectua căutări cu comanda SEEK:

USE contacte ORDER nr

SEEK 1

În exemplul precedent numărătorul de înregistrări se poziționează pe articolul a cărui câmp nr = 1. Dacă un astfel de articol nu există contorul rămâne neschimbat și funcția FOUND() returnează .F.. Comenzile SET EXACT ON/OFF modifică modul de operare a comenzii SEEK. Dacă este activă comanda SET EXACT OFF căutarea se va face numai după primele caractere ale câmpului cheie.

Crearea și exploatarea bazelor de date relaționale

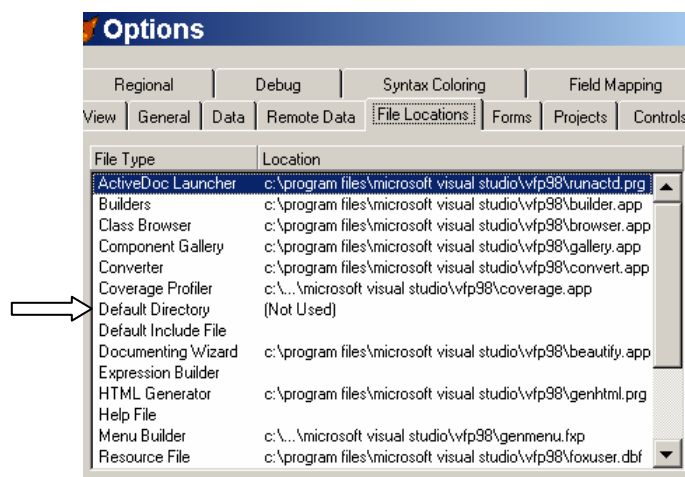
Se poate defini directorul implicit pentru fișiere. Pentru a defini un director implicit pentru sesiunea de lucru curentă se folosește comanda:

```
SET DEFAULT TO <nume_director>
```

De exemplu:

```
SET DEFAULT TO C:\UTILIZATORI\STUDENT\VASILE
```

Pentru a defini directorul implicit care să fie încărcat la fiecare pornire a VFP, din meniul, Tools/Options/File Locations, când se activează o fereastră în forma:



Comenzile se pot integra în fișiere de comenzi sau programe, care au facilități suplimentare față de facilitățile oferite de fereastra de comenzi. Oricum, oricare din succesiunea de comenzi (sau toate la un loc) care au fost lansate în fereastra de comenzi se pot salva într-un program. Un nou program se creează din *New/Program/New File*.

Se lansează în execuție cu comanda DO:

```
DO ProgramName1 | ProcedureName [IN ProgramName2] [WITH ParameterList]
```

Argumente:

- ProgramName1 – specifică numele programului de executat;

Dacă nu se include nici o extensie, VFP caută să execute în următoarea ordine:

ProgramName1.exe (versiunea executabilă)

ProgramName1.app (o aplicație)

ProgramName1.fxp (versiunea compilată)

ProgramName1.prg (programul)

Pentru a executa un program de tip meniu, o formă, o interogare, trebuie să i se includă și extensia (*.mpr*, *.spr*, sau *.qpr*).

ProcedureName – specifică numele procedurii de executat din programul curent.

IN ProgramName2 clauză care comunică VFP să caute procedura într-un fișier anume;

WITH ParameterList – specifică parametrii de transmis programului sau procedurii.

Se poate merge într-un program la apel recursiv la peste 128 de apeluri DO.

17. Expresii

Expresiile care pot fi construite cu ajutorul tipurilor de dată predefinite puse la dispoziție de mediul VFP (de exemplu în *Expresion Builder*) se găsesc documentate în MSDN, *Active Subset: Visual Fox Pro Documentation/Contents/Reference/Language Overview/Overview of the Language/Building Expressions*, sau dacă este instalată versiunea MSDN'98, atunci se poate încărca în *Internet Explorer* pagina *mk:@MSITStore:C:\Program%20Files\Microsoft%20Visual%20Studio\MSDN98\98VSpa\1033\foxhelp.chm::/html/conbuilding_expressions.htm*.

- Operatorii pe expresii de tip caracter sunt descriși în:

foxhelp.chm::/html/tblcharacter_operators.htm

- Operatorii pe expresii de tip dată și oră sunt descriși în:

foxhelp.chm::/html/tbldate_and_time_operators.htm

- Operatorii numerici:

foxhelp.chm::/html/tblnumeric_operators.htm

- Operatorii logici:

foxhelp.chm::/html/tbllogical_operators.htm

18. Lucrul cu funcțiile FVP – exemple de utilizare

ABS()

? ABS(-45) && Afișează 45

? ABS(10-30) && Afișează 20

? ABS(30-10) && Afișează 20

STORE 40 TO gnNumber1

STORE 2 TO gnNumber2

? ABS(gnNumber2-gnNumber1) && Afișează 38

CHR() – într-un program:

CLEAR

FOR nCOUNT = 65 TO 75

 ? nCount && Display numeric value

 ?? ' ' + CHR(nCount) && Display character

ENDFOR

DATE()

CLEAR

SET CENTURY OFF

Crearea și exploatarea bazelor de date relaționale

? DATE() && Afișează today's date without the century

SET CENTURY ON

? DATE() && Afișează today's date with the century

? DATE(1998, 02, 16) && Afișează a year 2000-compliant Date value

DATETIME()

tNewyear = DATETIME(YEAR(DATE()) + 1, 1, 1) && Next New Year

tToday = DATETIME()

nSecondstonewyear = tNewyear - tToday

CLEAR

? "There are " + ALLTRIM (STR(nSecondstonewyear)) ;

+ " seconds to the next New Year."

CLEAR

SET CENTURY ON

SET DATE TO AMERICAN

? DATETIME(1998, 02, 16, 12, 34, 56) && Afișează 02/16/1998 12:34:56 PM

STORE {^1998-03-05} TO gdBDate

DAY()

CLEAR

? CDOW(gdBDate) && Afișează Thursday

? DAY(gdBDate) && Afișează 5

? 'That date is ', CMONTH(gdBDate), STR(DAY(gdBDate),2)

DBUSED() && Returnează true (.T.) dacă baza de date specificată este deschisă.

DBUSED('Universitati')

DELETED()

DELETED([cTableAlias | nWorkArea]) Returnează o valoare logică ce indică dacă înregistrarea curentă este marcată pentru ștergere.

DMY() && expresie de tip caracter în formatul day-month-year

CLEAR

SET CENTURY OFF

? DMY(DATE())

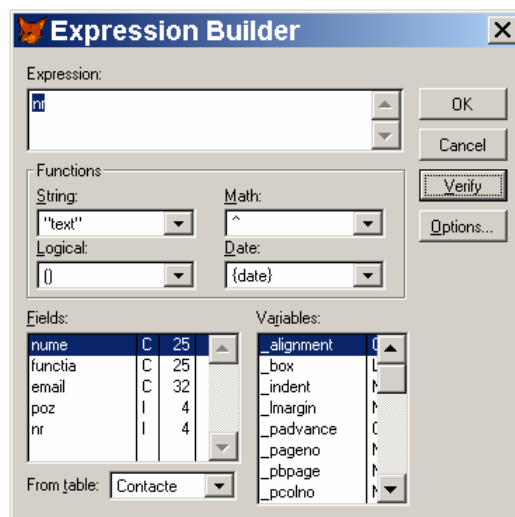
SET CENTURY ON

? DMY(DATE())

19. Constructorul de expresii (*Expression Builder*)

Constructorul de expresii este un instrument esențial în exploatarea unei baze de date. În elementele discutate el intervine în:

- definirea expresiilor pentru indecși;
- definirea regulilor de validare, mesajelor și valorilor implicite pentru câmpuri și înregistrări;
- definirea funcțiilor și expresiilor pentru noi câmpuri în interogări și vederi;
- definirea etichetelor de câmpuri la vederi;



Constructorul de expresii poate fi accesat din aplicațiile expert, ferestre, constructoare și wizard-uri. Permite definirea de expresii în care intervin câmpuri din tabele și vederi. Tipul expresiei construite trebuie să fie compatibil cu tipul acceptat de caseta de editare din care a fost încărcat. De exemplu, dacă a fost încărcat dintr-o casetă de editare în care se acceptă o valoare de tip șir de caractere (cum este cazul pentru *Fields Caption*) atunci valoarea expresiei construite trebuie să fie de tip șir de caractere.

O expresie poate fi:

- simplă (numele unui câmp);
- complexă (implicând de exemplu un IF *imediat*);

Pentru a construi expresii, se pot scrie în caseta de editare pentru expresii sau se pot selecta intrări din lista *drop-down* a funcțiilor pentru a se insera în caseta de editare.

Pentru operații cu șiruri de caractere sau câmpuri ce conțin șiruri de caractere, sunt utile funcțiile *ALLTRIM()*, *LTRIM()*, *RTRIM()*, *PADL()*, *PADR()*, *PADC()*, *SUBSTR()*, *LEFT()*, *RIGHT()*, *UPPER()*, *LOWER()*, *PROPER()*, *STR()*.

Să presupunem că avem șirul de caractere: " ABBA ". Atunci:

? *ALLTRIM*(" ABBA SS ") && va afișa "ABBA SS"

? *LTRIM*(" ABBA ") && va afișa "ABBA "

Crearea și exploatarea bazelor de date relaționale

? RTRIM(" ABBA ")	&& va afișa " ABBA"
? PADL(" ABBA ",12,"X")	&& va afișa "XXX ABBA "
? PADR(" ABBA ",12,"X")	&& va afișa " ABBA XXX"
? PADC(" ABBA ",12,"X")	&& va afișa "X ABBA XX"
? SUBSTR(" ABBA ",2,4)	&& va afișa " ABB"
? SUBSTR(" ABBA ",2)	&& va afișa " ABBA "
? LEFT(" ABBA ",3)	&& va afișa " A"
? RIGHT(" ABBA ",5)	&& va afișa "BA "
? LOWER(" ABBA ")	&& va afișa " abba "
? UPPER(LOWER(" ABBA "))	&& va afișa " ABBA "
? PROPER(" ABBA SS ")	&& va afișa " Abba Ss "
? STR(12+13)	&& va afișa " 25"
? STR(12+13,1)	&& va afișa "*"
? STR(12+13,2)	&& va afișa "25"
? STR(12.2)	&& va afișa " 12"
? STR(12.2,8,3)	&& va afișa " 12.200"
? STR(12.2,6,3)	&& va afișa "12.200"
? STR(12.2,5,3)	&& va afișa "12.20"

Pentru orice funcție selectată din *Expression Builder* sistemul VFP afișează pe linia de stare (*Status Bar*, ultima linie din fereastra aplicației VFP) informații cu privire la aceasta.

Din caseta *From table* a *Expression Builder* este permisă selectarea unei tabele din cele deschise (cu *USE*).

Caseta *Variables* permite utilizarea în definirea expresiei a variabilelor sistem, a tablourilor și variabilelor obișnuite anterior definite de utilizator.

Opțiunea *Verify* verifică sintaxa expresiei. Este însă doar o verificare formală, în timpul execuției nu este neapărat ca eroarea semnalată să fie reală. De exemplu definirea unei etichete în forma *Nume Contact* în caseta vederii *CautContact* nu este o eroare în execuție deși opțiunea *Verify* semnalează absența ghilimelelor: "*Nume Contact*".

Aplicație: se pot lista toate instituțiile au ca ultime două litere în acronim CN:

use institutii

browse font "Courier New" fields nume, acronim for upper(right(rtrim(acronim),2)) = "CN"

Rezultatul este în forma:

Institutii	
Nume	Acronim
Universitatea Tehnica Cluj-Napoca	UTCN
Universitatea de Stiinte Agricole si Medicina Veterinara Cluj-Napoca	USAMVCN

20. Programare

În general, orice ce poate fi făcut cu un program, se poate face manual dacă avem destul timp. De exemplu, dacă căutăm o persoană de contact în tabela *contacte* de exemplu Gheorghe Lazea, acest lucru poate fi făcut manual urmând secvența de instrucțiuni:

1. Din meniul *File* selectăm *Open*;
2. Din caseta *File of type* selectăm *Table*;
3. Selectăm tabela *Contacte.dbf* în lista fișierelor;
4. Din meniul *View* selectăm *Browse*;
5. Parcurgând conținutul tabelului, urmărind câmpul *Nume* identificăm înregistrarea pentru *Gheorghe Lazea*;

Din fereastra de comenzi, poate fi făcut același lucru scriind următoarele instrucțiuni:

USE Customer

LOCATE FOR Nume = "Gheorghe Lazea"

BROWSE

După ce am localizat înregistrarea putem să îi modificăm conținutul pentru a scrie de exemplu numele cu litere mari:

REPLACE nume WITH UPPER(nume)

sau să revenim la scrierea cu prima literă din nume mare:

REPLACE nume WITH PROPER(nume)

Instrucțiunile și comenzile pot fi integrate în *programe*. Acest fapt conferă unele avantaje:

- programele pot fi modificate și executate din nou;
- se pot executa programele din meniuri, forme și bare de instrumente;
- programele pot executa alte programe;

Un program în VFP se poate crea pe calea *New/Program, New File* sau din fereastra de comenzi cu comanda *modify command*.

Un program simplu care să schimbe toate numele la litere mari este:

use contacte

scan

replace nume with ;

upper(nume)

endscan

unde *scan* parcurge toate înregistrările din tabel și execută instrucțiunile între *scan* și *endscan* iar *;* indică că o comandă (comanda *replace*) se continuă pe linia următoare. Pe lângă comenzi, programele permit scrierea și de instrucțiuni (cum este cazul instrucțiunii *scan*).

Crearea și exploatarea bazelor de date relaționale

Pentru a restaura situația anterioară în fișier este suficient ca să modificăm programul înlocuind funcția *UPPER(ume)* cu funcția *PROPER(ume)*.

Pentru a executa o succesiune de comenzi și instrucțiuni în fereastra de comenzi, se introduc acestea în fereastra de comenzi (de exemplu cu *Copy* și *Paste*), se selectează (de exemplu de pe tastatură cu *Shift* și *săgeți*) după care se apasă enter (permite și execuția de instrucțiuni).

- pentru a crea un program: *New/Program, New File* sau comanda *modify command*;
- pentru a salva un program: *File/Save*;
- pentru a deschide un program: *File/Open/File type: program/Open* sau *modi comm <ume_program>*; se poate face și *modi comm ?* când se activează o fereastră de dialog;
- pentru a executa un program: *Program/Do...* sau cu comanda *do <ume_program>*;

Din perspectiva VFP există următoarele containere pentru date:

- variabile (elemente individuale de date stocate în RAM); pentru a crea o variabilă este suficient să precizăm numele variabilei și valoarea corespunzătoare; dacă nu există sau este de tip incompatibil, aceasta va fi creată când se distruge variabila anterioară cu același nume; alternativa pentru *store* este operatorul *=* cu semnificația cunoscută din limbajul C; variabilele pot fi publice, locale și private;

- de exemplu:

```
STORE DATE() TO local gdDate
```

```
STORE 50 TO gnNumeric
```

```
STORE 'Hello' TO gcCharacter
```

```
STORE .T. TO glLogical
```

```
STORE $19.99 TO gyCurrency
```

```
DIMENSION gaMyArray(2,2)
```

```
SET COMPATIBLE OFF
```

```
STORE 2 TO gaMyArray
```

```
CLEAR
```

```
DISPLAY MEMORY LIKE g*
```

- șiruri sau matrice (serii ordonate de elemente, stocate în RAM); se declară cu una din comenzile *DIMENSION* sau *DECLARE*; de exemplu:

```
DIMENSION ArrayName[5,2]
```

```
ArrayName[1,2] = 966789
```

```
? ArrayName[1,2]
```

- tabele și înregistrări; o înregistrare poate avea peste 255 de câmpuri;

Pentru lucrul cu tabele sunt utile comenzile *SCATTER*, *GATHER*, *COPY TO ARRAY*, și *APPEND FROM ARRAY*.

Instrucțiunea IF are sintaxa:

Instrucțiunea IF are sintaxa:

IF <expresie_logică> [THEN]

[<comenzi și instrucțiuni>]

[ELSE

<comenzi și instrucțiuni>]

ENDIF

Un exemplu de program cu instrucțiunea IF: să se afișeze înregistrările care îndeplinesc o condiție oarecare.

CLOSE DATABASES

OPEN DATABASE ('Università')

USE Contact

&& deschide tabela contacte

```
GETEXPR 'Introdu conditia de localizare ' TO gcTemp;
```

TYPE 'L' DEFAULT 'email' = ""

LOCATE FOR &gcTemp

&& Enter LOCATE expression

IF FOUND()

&& Este găsit?

$$DISPLAY$$

&& Dacă da, afișează înregistrarea

ELSE

&& Dacă nu e găsit

? 'Conditia ' + gcTemp + ' nu a fost gasita'

&& afișează un mesaj

ENDIF

USE

Instrucțiunea CASE are sintaxa:

DO CASE

CASE <expresie logică 1>

<comenzi și instrucțiuni>

[CASE <expresie_logică_2>

<comenzi și instrucțiuni>

...

CASE <expresie_logică_N>

<comenzi și instrucțiuni>]

[OTHERWISE

<comenzi și instrucțiuni>]

ENDCASE

Un exemplu de folosire al acesteia este următorul:

Crearea și exploatarea bazelor de date relaționale

STORE CMONTH(DATE()) TO luna	&& luna curentă
DO CASE	&& Începerea buclei case
CASE INLIST(luna, 'January', 'February', 'March')	
STORE 'Primul sfert al anului' TO rpt_titlu	
CASE INLIST(luna, 'April', 'May', 'June')	
STORE 'Al doilea sfert al anului' TO rpt_titlu	
CASE INLIST(luna, 'July', 'August', 'September')	
STORE 'Al treilea sfert al anului' TO rpt_titlu	
OTHERWISE	
STORE 'Al patrulea sfert al anului' TO rpt_titlu	
ENDCASE	&& Sfârșit buclă case
WAIT WINDOW rpt_titlu NOWAIT	&& afișare mesaj
Instrucțiuni de ciclare: SCAN, FOR, DO WHILE	
OPEN DATABASE ('Universitati')	&& Exemplu SCAN
USE Institutii in 1	&& Deschide tabela Institutii
USE Contacte in 2	&& Deschide tabela Contacte
CLEAR	
SCAN FOR institutii.nr = 1 and contacte.nr = 1	&& Permite și EXIT și LOOP (continuă)
? contacte.num, institutii.num, contacte.email	
ENDSCAN	
SET TALK OFF	&& Exemplu FOR
CLOSE DATABASES	
OPEN DATABASE ('Universitati')	
USE Contacte	
STORE 2 TO gnI	&& Valoare inițială
STORE 10 TO gnJ	&& Valoare finală
STORE 2 TO K	&& Pas
FOR gnCount = gnI TO gnJ STEP K	&& Permite și EXIT și LOOP (continuă)
GOTO gnCount	&& Mută pointerul de înregistrare
DISPLAY nume	&& Afișează numele
ENDFOR	&& Sau NEXT
CLOSE DATABASES	&& Exemplu DO WHILE
OPEN DATABASE ('Universitati')	
USE contacte	
SET TALK OFF	

DIMENSION A(4)

STORE 0 TO A

max = 0

DO WHILE .T.

&& Startul buclei DO WHILE

IF EOF()

&& Testează sfârșitul de fișier

EXIT

ENDIF

IF poz < 10

SKIP

LOOP

&& Trece la testarea din nou a condiției

ENDIF

A(nr) = A(nr) + 1

&& Numără ...; cel mult institutii.nr = 9

IF max < nr

&& Urmărește nr. maxim de instituții

max = nr

ENDIF

SKIP

ENDDO

&& Sfârșit buclă

CLEAR

? 'Statistica contactelor inregistrate mai puțin primele 10:'

FOR zz = 1 TO max

?? A(zz)

ENDFOR

21. Proceduri și funcții

În VFP procedurile și funcțiile au sintaxa în forma:

PROCEDURE <nume_procedură>

FUNCTION <nume_funcție>

<comenzi și instrucțiuni>

<comenzi și instrucțiuni>

ENDPROC

ENDFUNC

iar apelul acestora este în forma:

DO <nume_procedură>

DO <nume_funcție>

Următoarea procedură afișează un mesaj iar următoarea funcție arată o altă modalitate de a defini funcții și proceduri:

PROCEDURE myproc(cString)

FUNCTION plus2saptamani

MESSAGEBOX ("myproc" + cString)

PARAMETERS dDate

RETURN dDate + 14

ENDPROC

ENDFUNC

Crearea și exploatarea bazelor de date relaționale

În cazul în care procedura are parametrii, cum este cazul:

```
PROCEDURE procedura( dData, cSir, nOriTipar )
```

```
FOR nCnt = 1 to nOriTipar
```

```
    ? DTOC(dData) + " " + cSir + " " + STR(nCnt)
```

```
ENDFOR
```

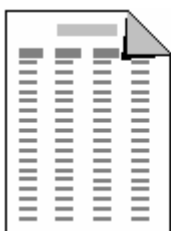
```
ENDPROC
```

atunci apelul se face în forma:

```
DO procedura WITH DATE(), "Salut Prieteni!", 10
```

22. Rapoarte și etichete

5 forme de prezentare a rapoartelor sunt frecvente:



raport pe coloane



raport pe linii



raport 1 la n



raport multi coloană



etichetă

Un raport se salvează pe disc cu extensia *.frx și are de asemenea asociat și un fișier cu extensia *.frt. Într-un raport se specifică câmpurile dorite, textul de imprimat și plasarea informației în pagină. Informația efectivă din raport se încarcă în momentul procesării raportului pentru imprimare și este conformă cu modificările care survin în tabelele și vederile folosite în întocmirea raportului.

Se poate crea un raport pe 3 căi: *Report Wizard* (pentru rapoarte dintr-o singură tabelă sau din mai multe tabele și/sau vederi), *Quick Report* (dintr-o singură tabelă sau vedere) și cu ajutorul lui *Report Designer* (rapoarte definite în întregime de utilizator).

Utilizarea lui Report Wizard

În *Project Manager* se selectează *Reports/New/Report Wizard*. Se selectează tipul de raport care se dorește a se crea și apoi se urmăresc instrucțiunile din ferestrele interogative.

Aplicație: realizarea unui raport cu persoanele contact din baza de date *Universitati*.

Rezolvare: se urmează pașii:

1. Se deschide proiectul *Universitati.pjx* (*File/Open...*);
2. Se încarcă baza de date *Universitati.dbc* (opțiunea *Modify*);
3. Se selectează tabulatorul *Documents* și de aici *Reports*;
4. Se acționează butonul *New...* și apoi se selectează *Report Wizard*;
5. Se selectează *One-to-Many Report Wizard* și se activează *Ok*;

Crearea și exploatarea bazelor de date relaționale

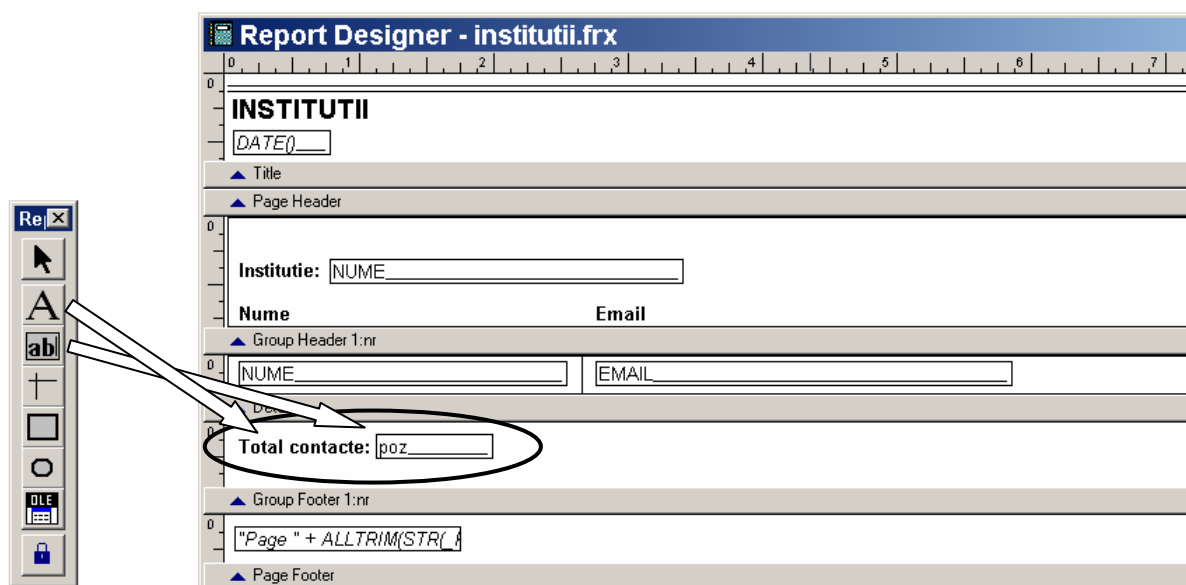
6. Se selectează din tabela părinte (*Institutii*) câmpul *nume*;
7. Se selectează din tabela fiu (*Contacte*) câmpurile *nume* și *email*;
8. Se acceptă relația *Institutii.nr = Contacte.nr* care leagă cele două tabele;
9. Se alege o regulă de ordonare în raport; fie aceasta după *acronim*;
10. Se alege o formă de prezentare; fie aceasta *Ledger*;
11. Se alege setarea de pagină; fie aceasta *Landscape*;
12. Se alege titlul raportului; fie acesta *institutii.frx*;
13. Se activează butonul *Preview* pentru a executa raportul.

Utilizarea lui *Report Designer*

Aplicație: totaluri pe categorii; pentru a realiza totaluri pe categorii este necesar să includem câmpuri numerice în raport, cum ar fi câmpul *poz* din tabela *contacte*.

Rezolvare: se modifică raportul *Institutii*:

1. Se deschide proiectul *Universitati.pjx* (*File/Open...*);
2. Se selectează tabulatorul *Documents* și de aici *Reports*;
3. Se acționează butonul *New...* și apoi se selectează *Report Wizard*;



4. Se modifică raportul cu ajutorul instrumentelor din *Report Controls* adăugând totalul după *poz* în *Group Footer*.

Realizarea unui raport dintr-o vedere și *Quick Report*

1. Se deschide proiectul *Universitati.pjx* (*File/Open...*);
2. Se încarcă baza de date *Universitati.dbc* (opțiunea *Modify*);
3. Se selectează tabulatorul *Documents* și de aici *Reports*;
4. Se acționează butonul *New...* și apoi se selectează *Report Wizard*;

Crearea și exploatarea bazelor de date relaționale

5. Se selectează *Report Wizard* și se activează *Ok*;
6. Se selectează vederea *contacte_institutii* cu structura: *contact.num*_e, *contact.email*, *institutii.num*_e (*contacte_institutii.num*_e_a, *contacte_institutii.email*, *contacte_institutii.email*);
7. Se lasă negrupate contactele;
8. Se sortează după *contacte_institutii.num*_e_a și apoi după *contacte_institutii.num*_e_b;
9. Se selectează boxa *Use display settings stored in the database*;
10. Se salvează raportul;
11. Se selectează raportul *Contacte_institutii* din tabulatorul *Docs (Documents)*;
12. Se apasă butonul *Preview...*;

CONTACTE INSTITUTII	
04/04/02	
Nume Contact	E-mail Contact
Nume Institutie	
Andrei Achimas Universitatea de Medicina si Farmacie "Iuliu Hatieganu" Cluj-Napoca	
Andrei Marga Universitatea "Babes-Bolyai" Cluj-Napoca	amarga@staff.ubbcluj.ro
Arpad Neda Universitatea "Babes-Bolyai" Cluj-Napoca	neda@staff.ubbcluj.ro
Doru Pamfil Universitatea de Stiinta, Arte si Medicina Valahia	dpamfil@usamvcluj.ro

Varianta 2

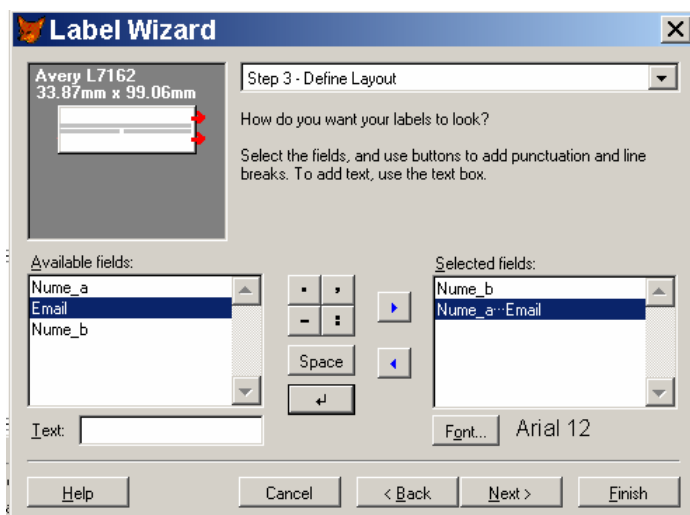
1. Se modifică vederea prin definirea ordinii de afișare *institutii.num*_e, *contacte.num*_e;
2. Se urmează pașii 1-6 ca la varianta 1;
3. Se grupează înregistrările după *num*_e_b;
4. Se alege stilul *Executive*;
5. Se sortează după *num*_e_a;
6. Se selectează boxa *Use display settings stored in the database*;
7. Se apasă *Finish* și se salvează raportul cu numele *contacte_institutii1*;

CONTACTE INSTITUTII	
04/04/02	
Nume Institutie	Nume Contact
	E-mail Contact
Universitatea "Babes-Bolyai" Cluj-Napoca	
	Andrei Marga amarga@staff.ubbcluj.ro
	Arpad Neda neda@staff.ubbcluj.ro
	Mircea Miclea mmiclea@staff.ubbcluj.ro
	Nicolae Bocsan nbocsan@staff.ubbcluj.ro
	Nicolae Paina

Etichete

Se urmează pașii:

1. Se selectează *Labels*, se apasă *New...*;
2. Se alege vederea *contacte_institutii*;
3. Se alege sistemul *metric*;
4. Se alege o formă de etichetă pe două coloane (de exemplu L7162);
5. Se definește forma etichetei (plasarea informațiilor în cadrul zonei etichetei);



6. Se definește fontul (de exemplu *Arial 12*);
7. Se salvează raportul;
8. Se vizualizează cu preview;

Universitatea "Babes-Bolyai" Cluj-Napoca
Andrei Marga amarga@staff.ubbcluj.ro

Universitatea "Babes-Bolyai" Cluj-Napoca
Arpad Neda neda@staff.ubbcluj.ro

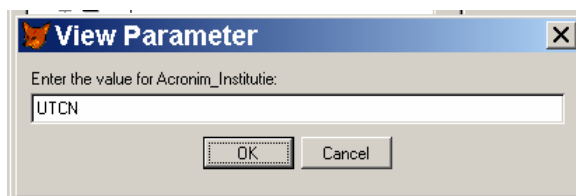
Universitatea "Babes-Bolyai" Cluj-Napoca
Mircea Miclea mmiclea@staff.ubbcluj.ro

Universitatea "Babes-Bolyai" Cluj-Napoca
Nicolae Bocsan nbocsan@staff.ubbcluj.ro

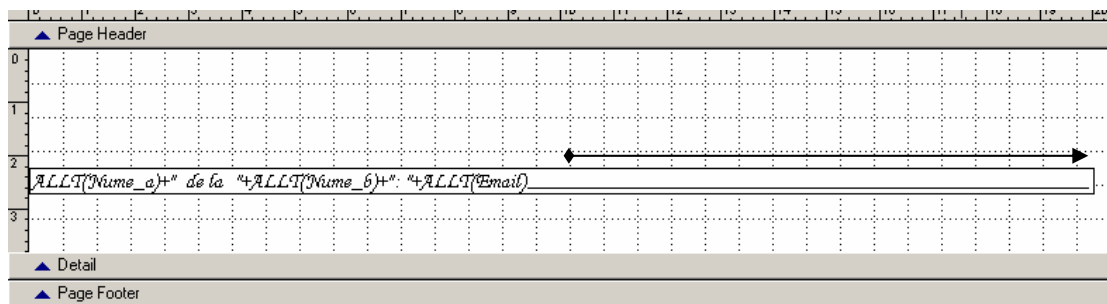
Crearea unei etichete dintr-o vedere cu parametru

Se poate folosi vederea *caut_contact* din baza de date universitati. Se urmează pașii:

1. În *Label Wizard* se alege vederea *caut_contact*;
2. Se alege sistemul *metric* și tipul *Avey EAL 04*;
3. Se definește forma (de exemplu: nume_a de la nume_b : Email);
4. Se alege fontul (de exemplu Monotype Corsiva, 12);
5. Se salvează eticheta *caut_contact*;
6. Se execută vederea de la butonul *Preview...* când se activează mesajul interogativ;



7. Pentru că spațiul alocat implicit de Wizard este insuficient pentru afișarea tuturor informațiilor (*email*) se alege modificarea raportului (*Modify*);
8. Se poate defini pagina *Landscape* (din *File/Page Setup/Print Setup/Orientation*);
9. Se mărește caseta alocată textului din Label Designer;



23. Macrosubstituție

Macrosubstituția este înlocuirea numelor cu variabile. În VFP se plasează operatorul `&` înaintea unei variabile pentru a folosi valoarea acestei variabile (care trebuie să fie un șir de caractere ce respectă sintaxa VFP) ca un nume. O comandă sau o funcție ce conține un nume se execută mai rapid ca una ce conține o macrosubstituție însă utilizarea macrosubstituțiilor conferă avantaje de flexibilitate în codul de executat.

Un exemplu de macrosubstituție poate fi crearea unei vederi folosind un cod SQL stocat într-o variabilă din care apoi poate fi chemat.

Codul se scrie într-un program (fișier de comenzi) sau se copiază în *Command*:

&& definirea vederii

con_inst_sql = "SELECT Contacte.numa, Contacte.email, Institutii.numa;

FROM universitati!institutii INNER JOIN universitati!contacte ;

ON Institutii.nr = Contacte.nr;

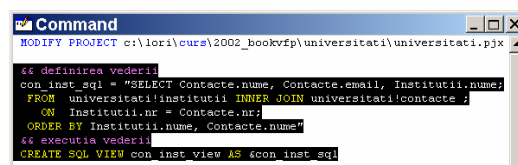
ORDER BY Institutii.numa, Contacte.numa"

&& execuția vederii

CREATE SQL VIEW con_inst_view AS &con_inst_sql

După execuție, în *Project Manager* apare vederea.

Aceasta se poate apoi vizualiza (Browse).



24. Formulare

Așa cum rapoartele permit facila tipărire a tabelelor, interogărilor și vederilor, formularele sunt o cale eficientă pentru afișarea, introducerea și editarea informațiilor din baza de date.

Se pot crea formulare interactive din tabele și vederi, utilizând wizard-ul, constructorul și aplicația expert Form Designer.

Form Wizard

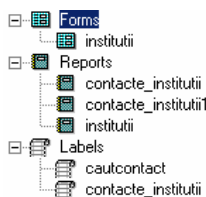
Din *Project Manager* din tabulatorul *Documents* se selectează *Forms* apoi *New* și *Form Wizard* după care se urmează instrucțiunile din ferestrele interogative următoare.



Aplicația 1. Să se creeze un formular pentru parcurgerea și modificarea datelor pentru instituțiile existente și pentru introducerea datelor pentru o nouă instituție în baza de date *universitati.dbc*.

Rezolvare. Se urmează pașii:

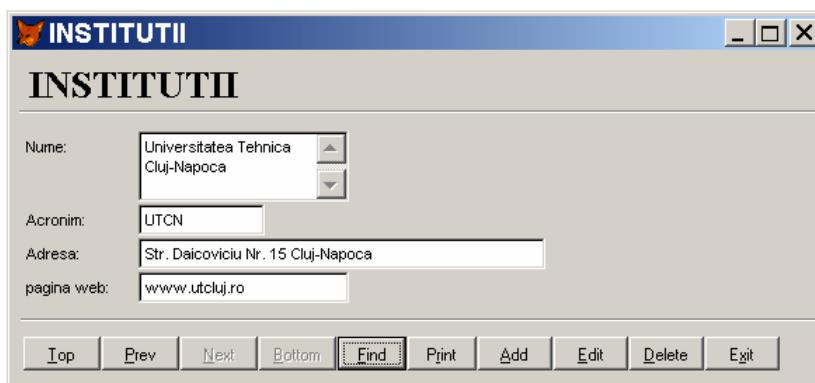
1. Se alege *Form Wizard* în fereastra de dialog *Wizard Selection*;
2. Se alege tabela *Institutii.dbc*; se includ câmpurile acesteia în caseta *Selected Fields* mai puțin câmpul autoincrement *universitati!institutii.nr*;
3. Se alege stilul formei; de exemplu *Embossed*; se alege tipul butoanelor de navigare; de exemplu *Text buttons*; se alege indexul după care să se facă ordonarea la afișare; de exemplu *acronim*;
4. Se salvează forma pentru a fi utilizată ulterior; se alege numele acesteia *institutii*; ea se va salva pe disc cu numele *institutii.scx*;



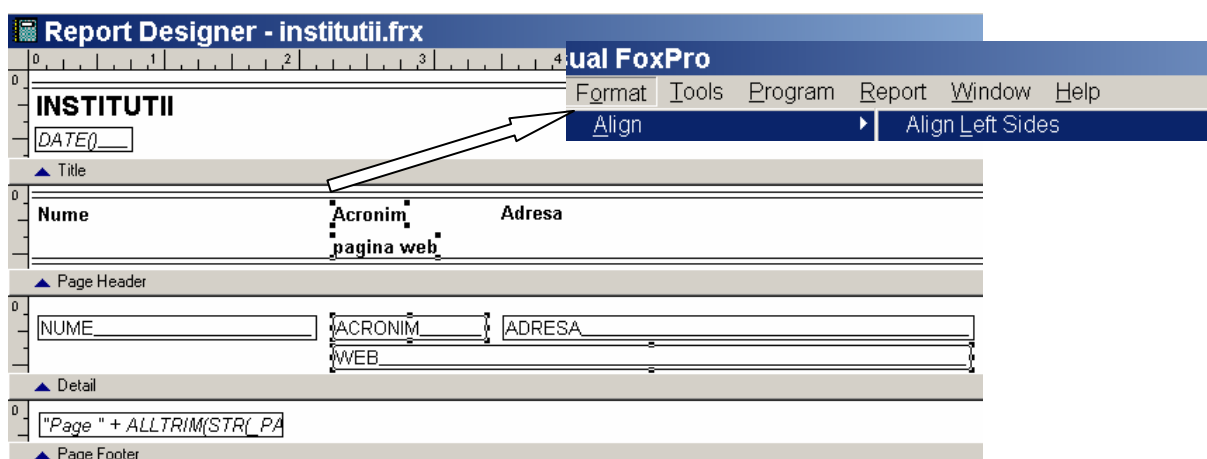
5. În *Project Manager* va apare în grupul *Forms* noua formă creată; se selectează forma *institutii* și se lansează în execuție (butonul *Run*).

Crearea și exploatarea bazelor de date relaționale

Noua formă creată conține 5 butoane pentru deplasare (*Top*, *Next*, *Prev*, *Bottom*, *Find*), un buton pentru imprimare (*Print*), 3 butoane pentru operații de intrare-ieșire (*Add* pentru adăugarea unei noi înregistrări, *Edit* pentru modificarea unei înregistrări existente și *Delete* pentru ștergere) și un buton pentru ieșire (*Exit*).



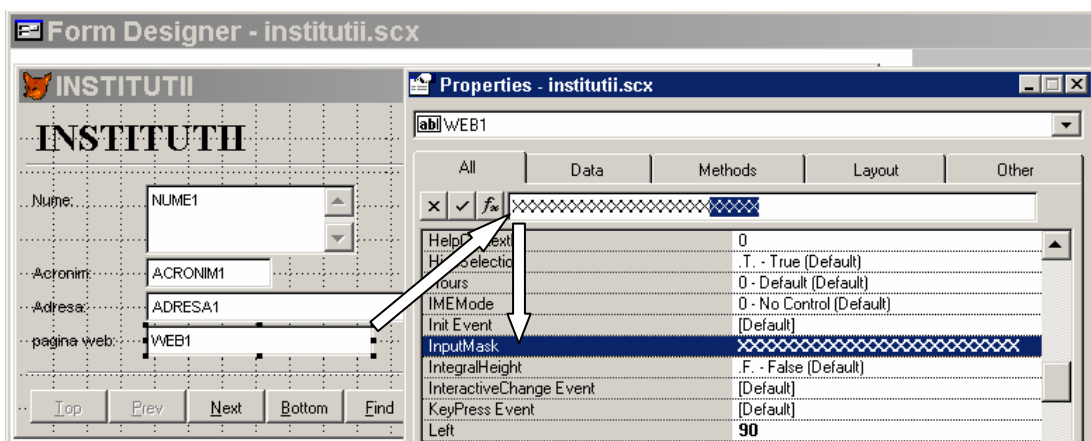
Pentru tipărirea informațiilor despre instituții se poate crea un raport *institutii.frx*; se poate folosi wizard-ul pentru generarea raportului incluzând câmpurile nume, acronim, adresa și web, selecta opțiunea de preluare a etichetelor pentru câmpuri din tabela *institutii.dbf* și apoi se poate modifica încât să se prezinte astfel:



Se poate folosi formularul pentru introducerea unei noi instituții (de exemplu Academia de Arte Vizuale „Ion Andreescu” Cluj-Napoca). Tentativa de a completa adresa web a acesteia este sortită eșecului dacă câmpul pentru pagina web are 20 de caractere. Sunt necesare atunci următoarele operațiuni:

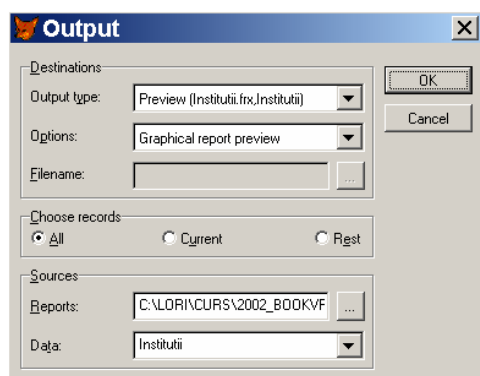
1. Se modifică structura tabelii *institutii.dbf* prin mărirea câmpului de la 20 la 25 de caractere (*Project Manager/Databases/Universitati/Tables/Institutii/Modify*);
2. Se modifică formularul *institutii.scx* (se mărește caseta de editare pentru pagina web) pe calea *Project Manager/Docs/Forms/Institutii/Modify*;
3. Se activează proprietățile casetei de editare WEB1 (*click dreapta* pe caseta corespunzătoare paginii web și apoi *Properties*);
4. Se modifică macheta de introducere a casetei prin inserarea a încă 5 "X";

Crearea și exploatarea bazelor de date relaționale



5. Pentru a executa formularul este necesar acum să închidem tabela *institutii.dbf* deschisă de sistem la modificarea structurii (execuția necesită accesul exclusiv la tabelă); se poate face acest lucru din *Data Session* sau din fereastra de comenzi *Close all* urmată de *Modi Project*);
6. Se execută formularul pe calea *Project Manager/Docs/Forms/Institutii/Run*;
7. Se pot completa acum datele;

8. Pentru inserarea datelor în tabela *institutii.dbf* se apasă butonul *Save* și apoi *Exit* când la o nouă execuție a formularului sau la o deschidere a tabelii *institutii* într-o fereastră *Browse* apare noua instituție înregistrată; dacă a fost definită funcția de autoincrementare pentru câmpul cheie *institutii.nr* acesta va avea valoarea generată automat în tabelă (vezi fereastra *Browse* pentru tabela *institutii*);
9. Tipărirea instituțiilor cu ajutorul formei *institutii.scx* se poate face acum prin selectarea raportului *institutii.frx*: *Project Manager/Docs/Forms/Institutii/Run/Print*;



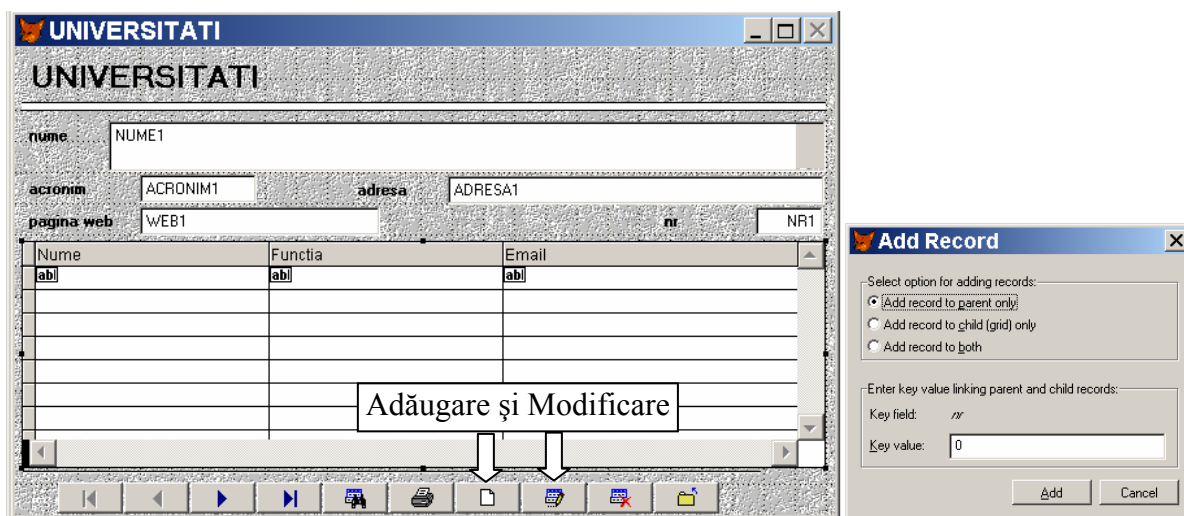
Nume	Acronim	Adresa
Academia de Arte Vizuale "Ion Andreescu" Cluj-Napoca	AAVIA	Piata Unirii nr. 31 Cluj-Napoca www.edu.ro/aavcn.htm
Universitatea "Babes-Bolyai" Cluj-Napoca	UBB	Mihail Kogalniceanu nr. 1B Cluj-Napoca www.ubbcluj.ro
Universitatea de Medicina si Farmacie "Iuliu Hatieganu" Cluj-Napoca	UMFIH	www.umfcluj.ro
Universitatea de Stiinta Agricole si Medicina Veterinara Cluj-Napoca	USAMVCN	Str. Manastur Nr. 3-5 Cluj-Napoca www.usamvcluj.ro
Universitatea Tehnica Cluj-Napoca	UTCN	Str. Daicoviciu Nr. 15 Cluj-Napoca www.utcluj.ro

Aplicația 2. Să se creeze un formular pentru parcurgerea și modificarea datelor simultan pentru instituțiile și persoanele de contact din baza de date *universitati.dbc*.

Rezolvare. Se urmează pașii:

1. Se alege *One-to-many Form Wizard* în fereastra de dialog *Wizard Selection*;
2. Se alege tabela părinte: *Institutii.dbc*; se includ câmpurile acesteia în caseta *Selected Fields* mai puțin câmpul autoincrement *universitati!institutii.nr*;
3. Se alege tabela fiu: *Contacte.dbf*; se includ câmpurile acesteia în caseta *Selected Fields* mai puțin câmpurile cheie străină (*Contacte.poz*) și autoincrement (*Contacte.nr*);
4. Se acceptă relația între tabele pe baza cheii primare *Institutii.nr* și străine *Contacte.nr*;
5. Se alege stilul formei; de exemplu *Stone*; se alege tipul butoanelor de navigare; de exemplu *Picture buttons*; se alege indexul după care să se facă ordonarea la afișare pentru înregistrările din tabela părinte; de exemplu *nr*;
6. Se salvează forma pentru a fi utilizată ulterior; se alege numele acesteia *universitati*; ea se va salva pe disc cu numele *universitati.scx*;

În *Project Manager* va apare în grupul *Forms* noua formă creată; se selectează forma *universitati* și se modifică ca mai jos (butonul *Modify*); apoi se lansează în execuție;



Formularul permite modificarea datelor pentru o instituție; prezența codului relației de integritate face ca modificările făcute valorii cheii primare (*Institutii.nr*) să se transmită și în tabela de contacte, fapt care se poate observa cu ajutorul formularului. De asemenea, modificările efectuate asupra înregistrărilor din tabela de contacte (mai exact modificările asupra cheii străine) sunt controlate de codul relației de integritate. O modificare a cheii străine la o valoare care nu se regăsește în tabela *institutii* este sortită eșecului.

Formularul permite adăugarea unei instituții sau a unei persoane de contact. Prezența relației de integritate interzice însă adăugarea înregistrărilor simultan în ambele tabele. Este posibilă adăugarea simultană numai prin renunțarea la codul relației de integritate.

Form Builder

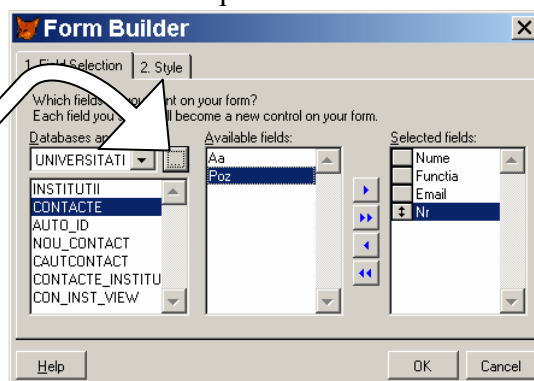
Constructorul de formulare (*Form Builder*) se poate activa pe următoarea succesiune de pași:

1. Se încarcă proiectul;
2. Se selectează opțiunea *Forms*;
3. Se apasă *New/New form*;
4. Din bara de instrumente a lui *Form Designer* se selectează *Form Builder* sau din meniul VFP, *Form/Quick Form*;

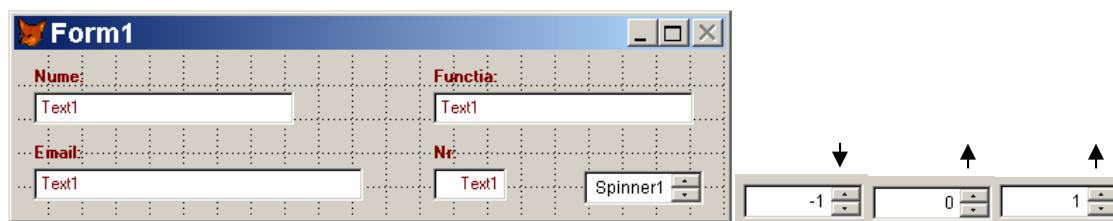
Aplicația 3. Să se construiască un formular pentru modificarea unei persoane de contact.

Rezolvare. Se urmează pașii:

1. Se încarcă *Form Builder*;
2. Se deschide *universitati.dbc* sau *contacte.dbf*;
3. Se selectează câmpurile din *contacte.dbf*;
4. Se alege stilul (de exemplu *Colorful*);
5. Se salvează formularul;
6. Se lansează în execuție;



Formularul va încărca din tabela *Contacte* prima înregistrare care va putea fi modificată după dorință. La închiderea formularului modificările efectuate se vor transmite în tabela *Contacte* (dacă nu a apărut conflict la modificarea cheii străine *contacte.nr*). Pentru a îmbunătăți formularul creat este necesar să adăugăm controale pe acesta cu ajutorul barei de instrumente *Form Controls*. Din aceasta se poate adăuga un control *Spinner* (*Spinner1*).



La lansarea în execuție a formularului se poate observa că săgețile controlului *Spinner1* (↑ și ↓) incrementează sau decrementează valoarea din caseta de editare a acestuia fără însă ca odată cu aceasta să se deplaseze pointerul înregistrării curente în tabela *Contacte* și să accesăm o altă înregistrare.

Este necesară setarea proprietăților controlului *Spinner1*. O posibilitate este ca mai jos:

- ControlSource **Contacte.poz**
- DownClick Event (**User Procedure**)
- SpinnerHighValue = **RECCOUNT()**
- SpinnerLowValue = **1**
- UpClick Event (**User Procedure**)

```
v = val(Form2.Spinner1.Text)
if v >= 1 and v <= recsize()
  goto v
  Form2.NUME1.Text1.refresh
  Form2.FUNCTIA1.Text1.refresh
  Form2.EMAIL1.Text1.refresh
  Form2.NR1.Text1.refresh
else
  return to master
endif
```

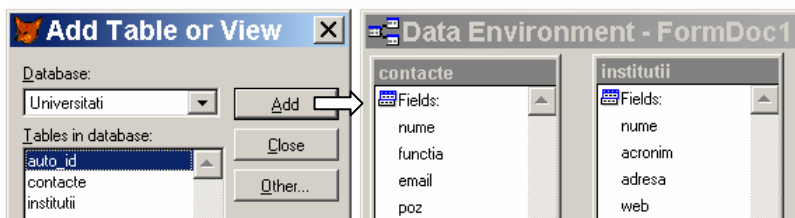
Form Designer

Pentru generarea manuală a formularelor sunt necesare bara de instrumente a lui *Form Designer* și a lui *Form Controls*.

Aplicația 4. Să se construiască un formular după modelul:

Rezolvare. Se urmează pașii:

1. Cu *Form Designer* se inserează tabelele *institutii* și *contacte* (butonul *Data Environment*);



2. Cu *Form Controls* se contruiesc butoanele, casetele de editare, listele combinate și etichete; se definește mărimea formularului, se fac alinierele;

3. Se creează câte un index după fiecare câmp din tabelele *contacte* și *institutii* (regular sau primar);
4. Se definesc proprietățile pentru fiecare *Combo* și *Text*;
 - a. Combo1
 - i. Click Event

select institutii

set order to nume

seek ALLTRIM(Form1.Combo1.Text)

Form1.Text1.Refresh

Form1.Text2.Refresh

Form1.Text3.Refresh

Form1.Text4.Refresh

Crearea și exploatarea bazelor de date relaționale

Form1.Text5.Refresh
select contacte
set filter to contacte.nr = institutii.nr
Form1.Combo6.Refresh
Form1.Combo7.Refresh
Form1.Combo8.Refresh
Form1.Combo9.Refresh
Form1.Combo10.Refresh

ii. RowSource: *Institutii.nume*

iii. Row Source Type: 6 – *Fields*

b. Combo2

i. Click Event

select institutii
set order to adresa
seek ALLTRIM(Form1.Combo2.Text)
Form1.Text1.Refresh
Form1.Text2.Refresh
Form1.Text3.Refresh
Form1.Text4.Refresh
Form1.Text5.Refresh
select contacte
set filter to contacte.nr = institutii.nr
Form1.Combo6.Refresh
Form1.Combo7.Refresh
Form1.Combo8.Refresh
Form1.Combo9.Refresh
Form1.Combo10.Refresh

ii. RowSource: *Institutii.adresa*

iii. Row Source Type: 6 – *Fields*

c. Analog *Combo3* și *Combo5*

d. Combo4

select institutii
set order to nr
seek val(ALLTRIM(Form1.Combo4.Text))
Form1.Text1.Refresh
Form1.Text2.Refresh
Form1.Text3.Refresh
Form1.Text4.Refresh
Form1.Text5.Refresh
select contacte
set filter to contacte.nr = institutii.nr
Form1.Combo6.Refresh
Form1.Combo7.Refresh
Form1.Combo8.Refresh
Form1.Combo9.Refresh
Form1.Combo10.Refresh

e. Command1 (All)

Crearea și exploatarea bazelor de date relaționale

i. Click Event

select contacte
set filter to
Form1.Combo6.Refresh
Form1.Combo7.Refresh
Form1.Combo8.Refresh
Form1.Combo9.Refresh
Form1.Combo10.Refresh

ii. Caption: *All*

- f. Text1: ControlSource *Institutii.nume*;
- g. Text2: ControlSource *Institutii.adresa*;
- h. *Text3, Text4, Text5* analog;
- i. Text7: ControlSource *Contacte.nume*;
- j. Text6: ControlSource *Contacte.email*;
- k. *Text8, Text9, Text10* analog;
- l. Combo7

i. Click Event

select contacte
set order to nume
seek ALLTRIM(Form1.Combo7.Text)
Form1.Text6.Refresh
Form1.Text7.Refresh
Form1.Text8.Refresh
Form1.Text9.Refresh
Form1.Text10.Refresh

ii. RowSource: *Contacte.nume*

iii. Row Source Type: *6 – Fields*

- m. *Combo6, Combo8* analog;
- n. Combo9

i. Click Event

select contacte
set order to poz
seek val(ALLTRIM(Form1.Combo9.Text))
Form1.Text6.Refresh
Form1.Text7.Refresh
Form1.Text8.Refresh
Form1.Text9.Refresh
Form1.Text10.Refresh

ii. RowSource: *Contacte.poz*

iii. Row Source Type: *6 – Fields*

- o. *Combo10* analog

5. Se salvează forma și se lansează în execuție când se obține o fereastră de tipul:

6. Se pot adăuga butoane pentru adăugare instituție și adăugare persoană de contact.
7. Pentru adăugare instituție: un buton Add care să aplice un *Append Blank* și un buton Save care să aplice un *Replace nume with AllTrim(Form1.Text1.Text)* și așa mai departe;
8. Pentru adăugare contact: un buton care să aplice un *Append Blank* urmat de *Form1.Text10.Text = Form1.Text5.Text* și un buton Save care să aplice un *Replace nume with AllTrim(Form1.Text7.Text)* și așa mai departe;
9. Noua formă creată este în conformitate cu relația de integritate; pentru protejarea cheilor la adăugare se poate seta proprietatea Enabled la .F.;
10. Valorile câmpurilor autoincrement de asemenea pot fi blocate la modificare cu ajutorul aceleiași proprietăți;
11. Cu ajutorul controlului *Image* și apoi a proprietății *Picture* se pot insera poze în format recunoscut de sistem (bmp, gif, jpg, etc.);

Următoarele controale sunt disponibile în mod implicit la crearea de formulare:



- *Label*: creează un control de tip etichetă;
- *Text Box*: creează un control casetă de editare cu introducerea de text pe o singură linie;
- *Edit Box*: creează un control casetă de editare cu introducerea de text pe mai multe linii;
- *Command Button*: creează un control de tip buton de comandă;
- *Command Group*: creează un control de tip grup de butoane de comandă;
- *Option Group*: creează un control de tip grup de butoane radio;

Crearea și exploatarea bazelor de date relaționale

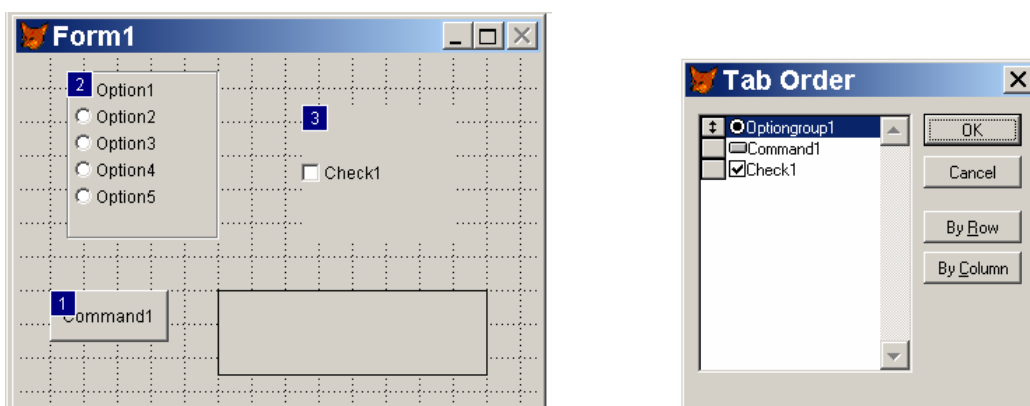
- *Check Box*: crează un control de tip casetă de selecție (opțiuni);
- *Combo Box*: control de tip listă ascunsă;
- *List Box*: control de tip listă vizibilă cu bare de defilare;
- *Spinner*: control de tip butoane incrementare/decrementare combinate cu o casetă de editare;
- *Grid Control*: un control de tip grid (vezi formularele one-to-many generate cu wizard-ul);
- *Image*: crează o imagine grafică;
- *Timer*: crează un control de timp;
- *Page Frame*: crează un grup de tabulatori;
- *Line* și *Shape*: permit trasarea de curbe și suprafețe pe formular;

Pentru alinierea și ordonarea controalelor pe formular este utilă bara de instrumente *Layout* (*View/Toolbars...*):



Form Designer conține și controale pentru definirea culorilor și formatarea formularelor. De asemenea, vizualizarea codului asociat unei proprietăți a formularului se poate face tot cu *Form Designer*.

Pentru a modifica ordinea de selecție a controalelor pe formular se selectează din meniu *View/Tab Order* când va apărea numărul de ordine al fiecărui control pe formular (numai pentru controalele care pot primi *focus*):



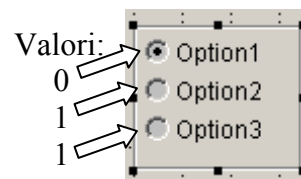
Schimbarea ordinii se face cu click pe control. Renumeroatarea începând cu poziția 1 se face cu dublu click pe controlul care se dorește a fi primul; apoi cu click pe al doilea, și așa mai departe. O altă posibilitate este de a seta ordinea pe baza unei liste. Din *Tools/Options/Forms/Tab ordering* se setează *By List* și apoi la *View/Tab Order* se activează o fereastră în care sunt așezate controalele într-o listă în ordinea de selecție (*Tab Order*).

Setarea ariei maxime pe ecran (implicit 640×480) a unui formular se face din *Tools/Options/Forms/Maximum design area*.

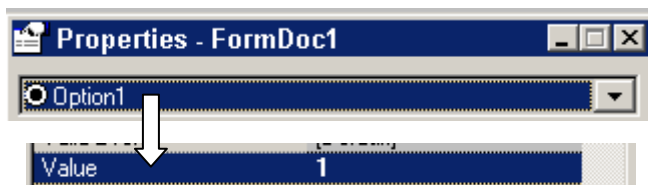
25. Controale

Controalele sunt mediul de bază pentru interacțiunea cu utilizatorul. Acestea se aplică pe formulare și pot avea asociate:

- valori (controlul buton de opțiune are valoarea implicită 0 dacă este selectat și valoarea implicită 1 dacă nu este selectat); dacă



numele formularului este *Form4* (proprietatea *Name*) iar numele grupului de butoane de opțiune este *OptionGroup1* atunci accesul la



valoare se face pe calea *Form4.OptionGroup1.Option1.Value*;

- câmpuri (proprietatea *ControlSource*);
- variabile

Aplicația1

Să se realizeze o aplicație cu butoane de opțiune (radio) care să afișeze un mesaj la selectarea unei opțiuni din caseta de opțiuni.

Rezolvare. Se pot urma pașii:

1. Din *Project Manager* se selectează *Documents (Docs)* apoi *Forms, New..., New Form*;
2. Se stabilește proprietatea *Caption* pentru formular (din *Form Designer, Properties Window*); fie *Caption Test*; acesta va fi afișat la execuția formularului;
3. Se stabilește proprietatea *Name* pentru formular; acesta va fi folosit pentru a identifica formularul în proceduri; fie *Name Form_Test*;
4. Se salvează formularul pe disc (*File/Save*); se stabilește un nume pentru formular; acesta va fi folosit pentru identificarea formularului în cadrul proiectului și va fi numele sub care acesta se salvează pe disc; fie acesta *Form3.scx*;
5. Se adaugă acum din *Form Controls* un control de tipul *Option Group*;
6. Se selectează controlul *Option Group* creat; dacă a fost închisă fereastra de proprietăți se activează din nou pe aceeași cale; acum avem 4 obiecte: formularul (cu proprietățile sale) și *OptionGroup1* (cu proprietățile sale) și două obiecte de tip butoane de opțiune (*Option1* și *Option2*); din caseta de proprietăți a lui *OptionGroup1* se selectează *Click Event* și se accesează această proprietate (*dublu click*);
7. În fereastra *Form_test.Click* care se activează și care conține codul procedurii care se execută la apăsarea unui buton de opțiune (vezi: *Object: Form_Test; Procedure: Click*) se introduce următorul cod:

Crearea și exploatarea bazelor de date relaționale

store "Starea butoanelor de optiune:" to xx

For i = 1 to ThisForm.OptionGroup1.ButtonCount

xx = xx + chr(13) + ThisForm.OptionGroup1.buttons[i].Caption +;

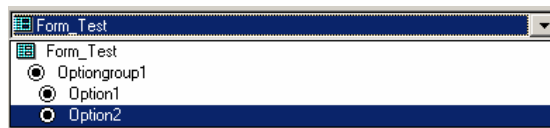
":" + str(ThisForm.OptionGroup1.Buttons[i].Value)

endfor

MessageBox(xx)

care pentru fiecare buton din grupul de butoane *OptionGroup1* extrage titlul acestuia (*Caption*) din tabloul (șirul) de butoane ale acestuia (*buttons*) și apoi valoarea (*Value*) pe care o convertește la șir de caractere (funcția *str(·)*) și le memorează în șirul de caractere *xx*; funcția *chr(13)* este folosită pentru trecerea la linie nouă; la sfârșit este afișat un mesaj cu conținutul șirului *xx* iar descriptorul *ThisForm* este folosit pentru a specifica formularul curent;

8. Se poate seta proprietățile *BackColor* (Red,Green,Blue) ale obiectelor (forma, grup de butoane, butoane); se pot seta acestea la alb (255,255,255), roșu (255,0,0), etc.; pentru accesul la butoane se poate merge pe calea ilustrată mai jos:



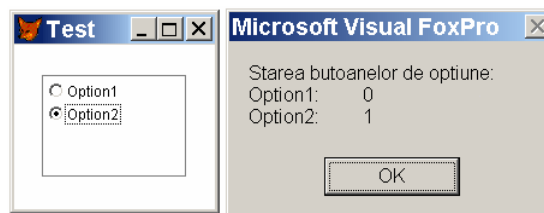
9. Pentru centrarea grupului de butoane pe formular se poate scrie următorul cod la proprietatea *Activate Event* (evenimentul de activare a formularului):

ThisForm.OptionGroup1.Left = (ThisForm.Width – ThisForm.OptionGroup1.Width)/2

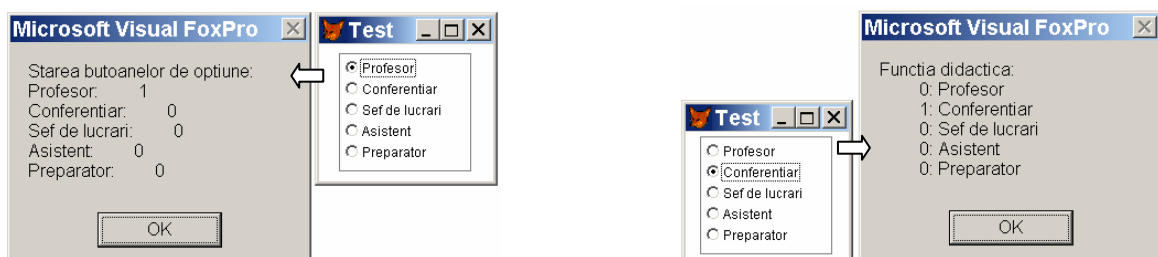
ThisForm.OptionGroup1.Top = (ThisForm.Height – ThisForm.OptionGroup1.Height)/2

10. Pentru ca la redimensionarea formularului la execuție să se centreze automat grupul de butoane în fereastra formularului, este necesară introducerea aceluiași cod și la evenimentul *Resize* al formularului;

11. Execuția formularului (butonul !) va duce la afișarea unui mesaj în forma:



12. Se pot acum modifica titlurile opțiunilor și chiar numărul acestora fără ca codul să sufere modificări; se poate modifica și mesajul implicit din fereastra de mesaj ca în figura:



Crearea și exploatarea bazelor de date relaționale

unde alinierea în fereastra de mesaj se poate face dacă se inversează ordinea de afișare:

store "Functia didactica:" to xx

For i = 1 to THISFORM.OptionGroup1.ButtonCount

xx = xx + chr(13) + str(THISFORM.OptionGroup1.Buttons[i].Value) +;

": " + THISFORM.OptionGroup1.buttons[i].Caption

endfor

MessageBox(xx)

Aplicația2

Să se modifice formularul *Form1* (Aplicația 4, pag. 54) astfel încât să permită:

12. Pentru adăugare instituție: un buton Add care să aplice un *Append Blank* și un buton Save care să aplice un *Replace nume with AllTrim(ThisForm.Text1.Text)* și așa mai departe;
13. Pentru adăugare contact: un buton care să aplice un *Append Blank* urmat de *Form1.Text10.Text = Form1.Text5.Text* și un buton Save care să aplice un *Replace nume with AllTrim(ThisForm.Text7.Text)* și așa mai departe;
14. Noua formă creată este în conformitate cu relația de integritate; pentru protejarea cheilor la adăugare se poate seta proprietatea *Enabled* la .F.;
15. Valorile câmpurilor autoincrement de asemenea pot fi blocate la modificare cu ajutorul aceleiași proprietăți;

Rezolvare. Se urmează pașii:

1. Se modifică setarea ariei maxime a formularelor la dimensiunea de 800×600 (*Tools/Options/Forms/Maximum design area*);
2. Se creează un nou formular în cadrul proiectului și se salvează cu un nume; fie acesta *form7.scx*;
3. Se selectează toate obiectele (controalele) de pe forma *form1.scx* cu ajutorul mouse-ului;
4. Se copiază în clipboard (*Edit/Copy*); se copiază pe forma *form7.scx* (*Edit/Paste*);
5. Se salvează din nou *form7*; se verifică existența tuturor procedurilor asociate evenimentelor de pe *form7*;
6. Se pot rearanja controalele pe formular;
7. Se includ tabelele *institutii* și *contacte* în mediul de lucru al formei *form7* (*Form Designer/Data Environment/Add Table or View/Add institutii, contacte*);
8. Pentru a permite adăugarea unei persoane de contact în prezența relației de integritate se asociază o valoare implicită pentru câmpul *nr* din tabela *contacte* (*contacte/Modify/Fields/nr/Field validation/Default value: 1*);
9. Se adaugă câte două controale de tip *command button* pentru fiecare tabelă pe formular;

Properties - form7.scx

Caption: Add
Click Event: (User Procedure):
 select institutii
 set filter to
 ThisForm.SetAll("Enabled",.F.)
 ThisForm.Command3.Enabled = .T.
 ThisForm.Text1.Enabled = .T.
 ThisForm.Text2.Enabled = .T.
 ThisForm.Text3.Enabled = .T.
 ThisForm.Text4.Enabled = .T.
 Append blank
 ThisForm.Refresh

Caption: Save
Enabled: .F. - False
Click Event: (User Procedure):
 ThisForm.SetAll("Enabled",.T.)
 ThisForm.Refresh

Caption: Save
Enabled: .F. - False
Click Event: (User Procedure):
 select institutii
 set order to nume
 seek
 ALLTRIM(ThisForm.Combo1.Text)
 select contacte
 set filter to contacte.nr = institutii.nr
 ThisForm.Refresh

Caption: Save
Enabled: .F. - False
Click Event: (User Procedure):
 select contacte
 set filter to
 ThisForm.SetAll("Enabled",.F.)
 ThisForm.Text6.Enabled = .T.
 ThisForm.Text7.Enabled = .T.
 ThisForm.Text8.Enabled = .T.
 ThisForm.Command5.Enabled = .T.
 Append Blank
 replace nr with val(ThisForm.Text5.Text)
 ThisForm.Refresh

Caption: Save
Enabled: .F. - False
Click Event: (User Procedure):
 ThisForm.SetAll("Enabled",.T.)
 ThisForm.Refresh

Click Event:

10. Se pot modifica evenimentele asociate controalelor

Combo1 – Combo5 după modelul lui *Combo1*;

11. Se salvează și se execută formularul, care permite acum și adăugarea unei instituții și/sau persoane de contact;

26. Controale și containere în FVP

Elementele puse la dispoziție de mediul VFP pentru dezvoltarea de aplicații se clasifică în controale și containere care pot fi vizuale și non-vizuale. Un container poate conține alte containere și controale. Sunt astfel două tipuri de clase: clasele *container* și clasele *control*. Orice container sau control care este folosit în aplicație reprezintă o instanțiere a clasei din care face parte și se numește *obiect* (de tip *container* sau *control*).

Dezvoltarea unei aplicații presupune alegerea elementelor potrivite pentru realizarea obiectivelor dorite. Cea mai rapidă cale de a dezvolta aplicații este prin folosirea elementelor standard puse la dispoziție de mediul VFP. Fiecare *obiect* (*container* sau *control*) are asociate *proprietăți* și *evenimente* accesabile atât în faza de construcție prin intermediul ferestrei de proprietăți asociate obiectului (*Properties Window*) cât și în faza de execuție (prin intermediul operatorului "."). Următoarea schemă clasifică și ierarhizează obiectele VFP:

Crearea și exploatarea bazelor de date relaționale

- Obiecte VFP:
 - Controale:
 - Vizuale (pot fi vizibile la execuție):
 - Check Box ()
 - Combo Box ()
 - Command Button ()
 - Control (poate conține orice control)
 - Edit Box ()
 - Header (creează o casetă *Header* pentru o coloană într-un control de tip *grid*)
 - Hyperlink () creează o legătură către un *Document Activ* prin intermediul unui Container de Documente Active (ex: Microsoft Internet Explorer), legătură dată printr-un URL (*uniform resource locator*)
 - Image () creează o legătură către o imagine care este afișată în format BMP
 - Label () creează o etichetă care poate afișa un text
 - Line () creează un control care poate afișa o linie
 - List Box () poate afișa o listă de valori
 - OLE Bound Control () poate afișa un obiect de tip *general* într-un câmp dintr-un tabel (de exemplu provenit din Word sau Excel)
 - OLE Container Control () poate afișa un obiect de tip *general* în aplicație (de exemplu provenit din Word sau Excel)
 - OptionButton (adaugă un buton de opțiune într-un container de tipul *Option Button Group*)
 - Shape () creează un control care poate afișa o formă geometrică
 - Spinner () creează un control care poate afișa o castă de editare de tip text pentru o valoare și două butoane pentru parcurgerea valorilor unui domeniu de valori asociat casetei de tip text
 - Text Box () creează un control care poate afișa o casetă de editare de tip text
 - Non-vizuale:
 - Active Doc (Creează un *document activ* care poate fi găzduit de un container de documente active ca *Microsoft Internet Explorer*)
 - Custom (poate conține orice control, Pageframe, container, custom)
 - Project Hook (poate conține fișiere și servere)
 - Timer () creează un control de timp
 - Container:
 - Vizuale:
 - Container (poate conține orice control, )
 - Form (poate conține Pageframe, orice control, containere, Custom)
 - Grid (poate conține *coloane grid*; *coloanele grid* fac legături la orice tip de obiecte exceptând formularele, seturile de formulare, barele de instrumente, controalele de timp și alte *coloane de grid*); cu ajutorul *grid-urilor* () se pot afișa datele în linii și coloane și este similar cu apariția lor într-o fereastră Browse
 - Column (creează o *coloană* într-un *grid*)
 - Page (o *pagină* poate conține orice controale, containere, Custom)
 - Toolbar (poate conține orice control, pagini, containere)
 - Option Button Group (poate conține butoane de opțiune, )
 - Command Button Group (poate conține butoane de comandă, )
 - Non-vizuale:
 - Form Set (poate conține formulare și bare de instrumente)
 - PageFrame (poate conține *pagini*)

Crearea și exploatarea bazelor de date relaționale

Toate obiectele VFP au asociate următoarele evenimente (prin proceduri):

- Init: este executată atunci când obiectul este creat;
- Destroy: este executată atunci când obiectul este dealocat din memorie;
- Error: este executată atunci când o eroare apare la execuția procedurilor din obiect;

Pentru accesarea obiectelor VFP din aplicație în timpul execuției sunt utile următoarele proprietăți:

- Parent: containerul care conține obiectul care referă procedura; de exemplu pentru formularul *Test.scx* proprietatea *Parent* apelată în procedura *Option1.Click* va returna o referință către containerul acestui obiect (*OptionGroup1*);
- This: obiectul curent;
- ThisForm: formularul curent;
- ThisFormSet: FormSetul curent;

Exemplu:

1. Se selectează obiectul *Option1* din formularul *Test*;
2. În caseta procedurii *Option1.Click* (*Click Event*) se introduce `MessageBox("Container: "+This.Parent.Name + " Formularul: " + ThisForm.Caption);`
3. Se execută formularul;

Un control poate fi legat de datele din tabele sau din variabilele din memorie pe baza proprietății *ControlSource*. Schematizarea efectelor valorilor atribuite proprietății *ControlSource* asupra controalelor:

- casetă de validare:
 - *ControlSource* = câmp de tabelă:
 - valorile NULL, valorile logice (.T. și .F.) și valorile numerice 0, 1 și 2 determină: selectarea, deselectarea sau dezactivarea casetei de validare pe măsură ce indicatorul de înregistrări parcurge tabela;
 - exemplu: să se construiască o tabelă ce conține un câmp cu valori numerice de 0, 1 și 2 și să se construiască un formular care conține o casetă de validare legată cu *ControlSource* de acest câmp și să se execute formularul;
- coloană:
 - *ControlSource* = câmp de tabelă:
 - utilizatorul editează direct valorile câmpului odată cu editarea valorilor din coloană; proprietatea se extinde la întreaga grilă cu ajutorul proprietății *RecordSource* a grilei;
 - exemplu: proprietatea *RecordSource* de la formularul *Universitati*, obiectul *grid1*;

Crearea și exploatarea bazelor de date relaționale

- casetă combo sau list:
 - ControlSource = variabilă:
 - valoarea aleasă de utilizator este păstrată în variabilă;
 - exemplu: se adaugă un control de tip *List (List1)* la formularul *Test*; la evenimentul de activare a formularului se adaugă instrucțiunile:

dimension a(10)

public i

store 2 to i

store " " to a

ThisForm.List1.additem(str(1))

la evenimentul DblClick al listei *List1* se introduc instrucțiunile:

ThisForm.List1.additem(str(i))

i = i + 1

se execută formularul; la dublu click se va insera câte un element în listă;

- buton de opțiune:
 - ControlSource = câmp numeric:
 - în câmp va fi inserată valoarea 0 (dacă este selectat butonul) sau 1 (dacă nu este selectat butonul);
 - ControlSource = câmp logic:
 - în câmp va fi inserată valoarea .T. (dacă este selectat butonul) sau .F. (dacă nu este selectat butonul);
 - dacă indicatorul de înregistrări parcurge tabela, valoarea butonului de opțiune se modifică pentru a reflecta noua valoare a câmpului;
 - exemplu: să se insereze un control de tip buton de opțiune pentru înregistrările șterse dintr-o tabelă;
- grup de opțiuni:
 - ControlSource = câmp de tip caracter:
 - se păstrează în câmp titlul butonului selectat;
 - opțiunea nu este valabilă și pentru butoane individuale de opțiuni;
- spinner:
 - ControlSource = câmp sau variabilă numerică:
 - caseta de incrementare afișează și scrie valori numerice din/în câmpul sau variabila asociate;
 - exemplu: formularul din aplicația 3 (pag. 53);
- text sau edit:

Crearea și exploatarea bazelor de date relaționale

- ControlSource = câmp:
 - în casetă este afișată valoarea câmpului din tabelă;
 - modificările efectuate sunt inserate pe loc în tabelă;

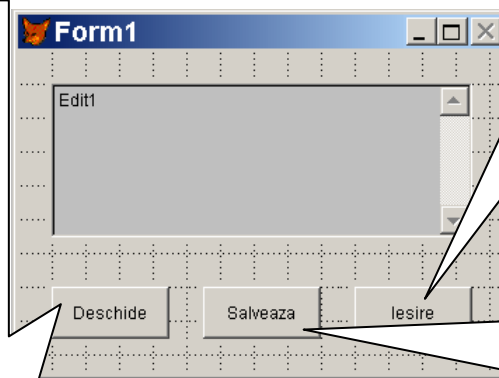
Aplicația3

Să se folosească controlul Casetă de Editare pentru a edita un fișier text.

Rezolvare:

Se creează o formă ca în figură:

Click Event:
create cursor textfile;
(filename c(35), mem m)
append blank
replace textfile.filename with;
getfile("txt")
if empty(textfile.filename)
return
endif
append memo mem from;
(textfile.filename) overwrite
ThisForm.Edit1.ControlSource;
= "textfile.mem"
ThisForm.Refresh
ThisForm.Command2.Enabled;
= .T.



Click Event:
Release ThisForm

Click Event:
copy memo textfile.mem;
to (textfile.filename)
Enabled:
.F. - False

Cu ajutorul grid-urilor se pot crea formulare care să opereze cu informații din mai multe tabele relatate cu relații de tipul 1 la n, ca în exemplul:

Customer name: Customer ID:

Order	Date	Ship To	Total
10062	10/15/93	Alfred's Futterkiste	\$900.40
10643	09/12/95	Alfred's Futterkiste	\$1,086.00
10692	10/21/95	Alfred's Futterkiste	\$878.00

Items for order 10062:

Item	Product	Qty.	Price
1	Chef Anton's Gumbo Mix	8.000	\$14.90
2	Manimup Dried Apples	12.000	\$37.10
3	Pâté chinois	20.000	\$16.80

Time format: ☒ 12-hour ☐ 24-hour

System clock: **Thursday April 18, 2002 2:27:25**

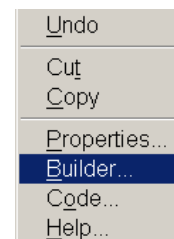
Cu ajutorul controlului de timp se pot construi formulare ca în exemplele:

27. Constructoarele de controale și containere

Mediul VFP pune la dispoziția utilizatorului un set de constructoare pentru setarea proprietăților controalelor și containerelor. Astfel există: *Combo Box Builder* (pentru liste ascunse), *Command Group Builder* (pentru grupuri de butoane de comandă), *Edit Box Builder* (pentru casete de tip Edit), *List Box Builder* (pentru liste), *Option Group Builder* (pentru casete de opțiuni), *Text Box Builder* (pentru casete de tip Text), *AutoFormat Builder* (pentru grupuri de controale).

Pentru a accesa un constructor:

1. Se plasează controlul pe formular din bara de instrumente *Form Controls*;
2. Se selectează controlul; se apasă click dreapta;
3. Se selectează opțiunea *Builder...*;
4. Se aleg opțiunile dorite din aplicația expert corespunzătoare controlului sau containerului selectat;

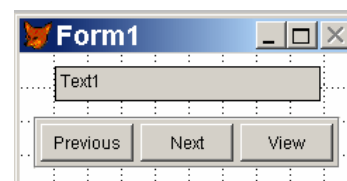


Aplicația 1.

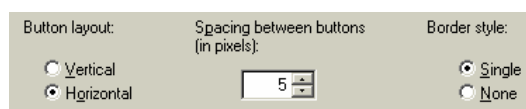
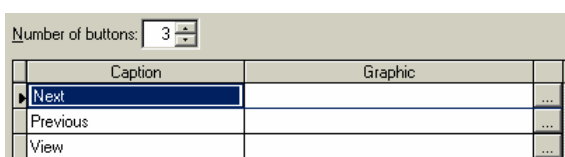
Formular cu 3 butoane de comandă care să deschidă și să parcurgă tabela *contacte*. La apăsarea unuia dintre butoane să se activeze o fereastră de mesaj cu conținutul înregistrării curente. Să se adauge apoi o casetă de tip Text care să conțină numele persoanei de contact selectate fără a permite modificarea.

Rezolvare. Se pot urma pașii:

1. Se generează un nou formular (*Forms/New.../New Form*);
2. Se adaugă un control *Text* și un container *CommandGroup*;
3. Se folosește aplicația expert *Text Box Builder* pentru a seta proprietăți pentru controlul *Text1*;



4. Odată selectată tabela *contacte*, aceasta va fi automat inclusă în mediul de lucru cu date al formularului; se poate vedea acest fapt pe calea *View/Data Environment ...*; de asemenea, în caseta *Field name*: se poate acum selecta orice câmp din tabelă; se lasă *contacte.nume*;
5. Se folosește aplicația expert *Command Group Builder* pentru a seta proprietăți pentru containerul *CommandGroup1*;



Crearea și exploatarea bazelor de date relaționale

6. Se poate lansa în execuție formularul, când se observă că butoanele din container nu au asociate evenimente: apăsarea butoanelor nu produce acțiuni;
7. Se adaugă evenimente:

- a. pentru formular: *Activate Event*:

```
if not eof()
  thisform.commandgroup1.enabled = .T.
endif
```

- b. pentru commandgroup1 doar proprietatea *commandgroup1.enabled = .F.*

- c. pentru command1, procedura Command1.Click:

```
if recno() = 1
  This.Enabled = .F.
else
  skip -1
  ThisForm.Text1.Refresh
  ThisForm.CommandGroup1.Command2.Enabled = .T.
endif
```

- d. pentru command2, procedura Command2.Click:

```
if recno() = reccount()
  This.Enabled = .F.
else
  skip 1
  ThisForm.Text1.Refresh
  ThisForm.CommandGroup1.Command1.Enabled = .T.
endif
```

- e. pentru command3, procedura Command3.Click:

```
mesaj = alltrim(ume) + chr(13) + alltrim(funcția) +;
chr(13) + alltrim(email) + chr(13) + "nr=" + alltrim(str(nr)) +;
chr(13) + "poz=" + alltrim(str(poz))
MessageBox(mesaj,64)
```

8. Se execută formularul; acționarea butoanelor de comandă produce acum evenimentele dorite.

Aplicația 2.

Formular care să permită adăugarea unei persoane de contact după o machetă predefinită.

Rezolvare. Se creează un nou formular. Se urmează pașii:

1. se adaugă pe formular un control de listă ascunsă, un container de tipul butoane de opțiune, 4 casete de tip text și un buton de comandă;
2. se folosește aplicația expert *Combo Box Builder* pentru a seta proprietăți pentru Combo1; din baza de date *universitati* și tabela *contacte* se alege câmpul *nume* pentru a i se lega valorile cu lista ascunsă; la stilul listei se alege *Drop-down list*;

Crearea și exploatarea bazelor de date relaționale

3. se folosește aplicația expert *Option Group Builder* pentru a genera containerul OptionGroup1; se introduce numărul de butoane (6) și denumirile acestora (vezi figura); se alege poziționarea *verticală* a butoanelor, și *spațiere* de 5 puncte între acestea;
4. se folosește aplicația expert *Text Box Builder* pentru a seta proprietăți pentru controalele Text2-Text4; acestea se setează astfel: aliniament: Text2: *right*; Text3: *left*, Text4: *center*; de asemenea, pentru Text4: 1. Format: ☒ *Make read only*; Input Mask: @;
5. se setează manual celelalte proprietăți și evenimentele pentru controale după cum urmează:

6. procedura Combo1.Click:

```
thisform.optiongroup1.Visible = .T.
```

```
thisform.text1.Visible = .T.
```

7. proprietatea OptionGroup1.Visible: *.F. – False*;

8. procedura Text1.GotFocus:

```
thisform.text2.visible = .T.
```

9. proprietatea Text1.MaxLength: =fsize('nume','contacte')

10. proprietatea Text1.Visible: *.F. – False*;

11. proprietatea Text2.InteractiveChange:

```
if "@" $ this.text
```

```
    thisform.text4.Visible = .T.
```

```
    thisform.text3.Visible = .T.
```

```
    v_email = substr(this.text, 1,at("@",this.text)-1)
```

```
    this.refresh
```

```
    thisform.text3.MaxLength = fsize('email','contacte');
```

```
    - len(alltrim(v_email))
```

```
    thisform.text3.setfocus
```

```
endif
```

```
if len(alltrim(this.text)) = this.maxlength - 2
```

```
    MessageBox("Lungimea maxima pentru acest camp a fost atinsa."+;
```

```
    chr(13)+"Pentru a putea adauga noi caractere mariti dimensiunea campului in tabela")
```

```
endif
```

12. proprietatea Text2.MaxLength: =fsize('email','contacte');

13. proprietatea Text2.ControlSource: *v_email*

14. proprietatea Text2.Visible: *.F. – False*;

15. proprietatea Text4.Visible: *.F. – False*;

16. procedura Text3.GotFocus:

```
thisform.command1.visible = .T.
```

17. procedura Text3.InteractiveChange:

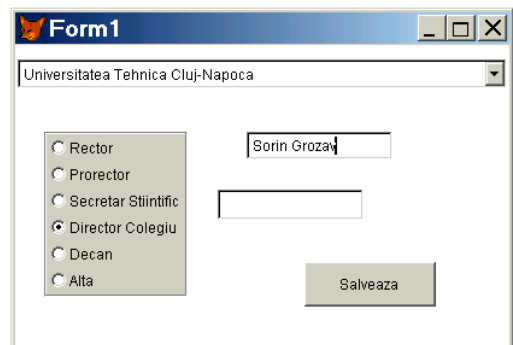
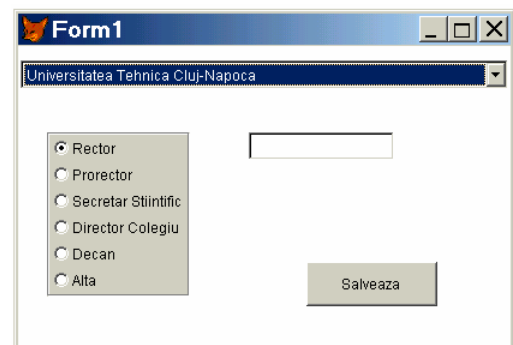
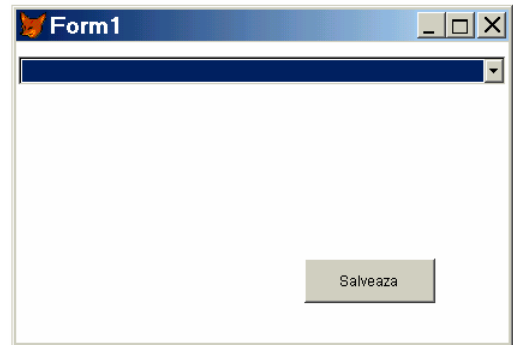
```
if len(alltrim(this.text)) = this.maxlength - 1
```

```
    MessageBox("Lungimea maxima pentru acest camp a fost atinsa."+;
```

```
    chr(13)+"Pentru a putea adauga noi caractere mariti dimensiunea campului in tabela")
```

```
endif
```

18. proprietatea Text3.Visible: *.F. – False*;



Crearea și exploatarea bazelor de date relaționale

19. proprietatea Command1.Caption: *Salveaza*;

20. procedura Command1.Click:

```
if (len(alltrim(thisform.text1.text))>3)and;  
(len(alltrim(thisform.text2.text))>0)and;  
(len(alltrim(thisform.text3.text))>0)  
    select contacte  
    append blank  
    replace nr with institutii.nr,;  
    nume with thisform.text1.text,;  
    email with alltrim(thisform.text2.text)+;  
    thisform.text4.text+alltrim(thisform.text3.text)  
    for i = 1 to thisform.optiongroup1.buttoncount  
        if thisform.optiongroup1.buttons[i].value = 1  
            replace functia with thisform.optiongroup1.buttons[i].Caption  
        endif  
    endfor  
else  
    this.visible = .F.  
    MessageBox("Completeaza corect formularul!")  
    if len(alltrim(thisform.text3.text))=0  
        thisform.text3.visible = .F.  
        thisform.text2.setfocus  
    endif  
    if (len(alltrim(thisform.text2.text))=0)or;  
        (len(alltrim(thisform.text1.text))<4)  
        thisform.text1.setfocus  
    endif  
endif
```

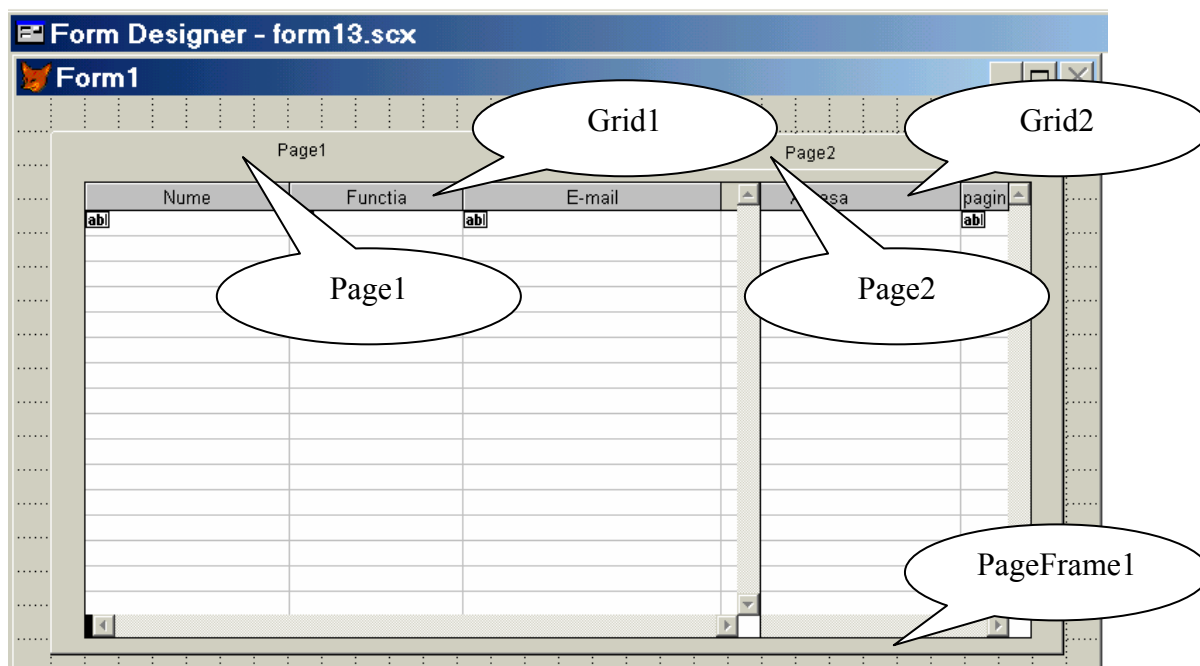
21. Se modifică formularul adăugându-se etichetele corespunzătoare;

22. La execuția formularului, dacă se introduc date incorecte (vezi procedura *Command1.Click*) atunci va apare o fereastră de mesaj și utilizatorul este întors la completarea casetei cu date incorecte:

Aplicația 3.

Să se folosească containerele *PageFrame* și *Grid* pentru vizualizarea datelor din baza de date *universitati*.

Rezolvare: se poate construi un formular după modelul:



Se urmează pașii:

1. Se creează un nou formular (Form1);
2. Se adaugă un container *PageFrame* (PageFrame1);
3. Se adaugă un container *Grid* (Grid1);
4. Se adaugă al doilea container *Grid* (Grid2);
5. Se selectează din fereastra de proprietăți containerul Grid1 (pentru instituții);
6. Se lansează aplicația expert *Grid Builder* pentru definirea proprietăților containerului Grid1; în tabulatorul *Grid Items* se selectează tabela *institutii* și din caseta *Available fields* se includ în caseta *Selected fields* câmpurile: *Nume*, *Acronim*, *Adresa*, *Web*;
7. Se selectează din fereastra de proprietăți containerul Grid2 (pentru contacte);
8. Se lansează aplicația expert *Grid Builder* pentru definirea proprietăților containerului Grid1; în tabulatorul *Grid Items* se selectează tabela *contacte* și din caseta *Available fields* se includ în caseta *Selected fields* câmpurile: *Nume*, *Functia*, *Email*;
9. În tabulatorul *Relationship* în caseta *Key field in parent table*: se selectează *Institutii.nr* iar în caseta *Related index in child table*: se selectează câmpul *nr*;
10. Se setează manual proprietățile și evenimentele controalelor și containerelor:

10.1. evenimentul Form1.Activate:

ThisForm.Grid1.Visible = .T.;

10.2. evenimentul Page1.Click:

ThisForm.Grid2.Visible = .F.

ThisForm.Grid1.Visible = .T.

10.3. evenimentul Page2.Click:

ThisForm.Grid1.Visible = .F.

ThisForm.Grid2.Visible = .T.

Crearea și exploatarea bazelor de date relaționale

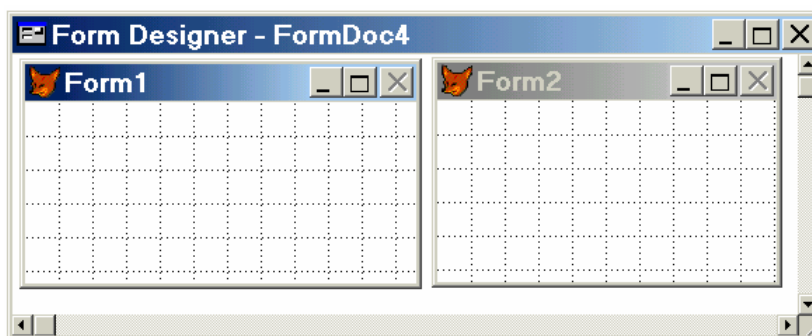
- 10.4. proprietatea Grid1.Visible = .F.
- 10.5. proprietatea Grid2.Visible = .F.
- 10.6. proprietatea Grid1.ReadOnly = .T.
- 10.7. proprietatea Grid2.ReadOnly = .T.

Aplicația 4. Seturi de formulare

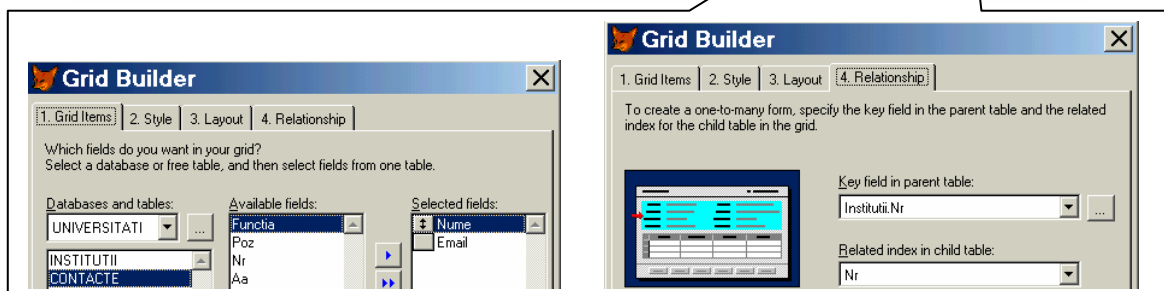
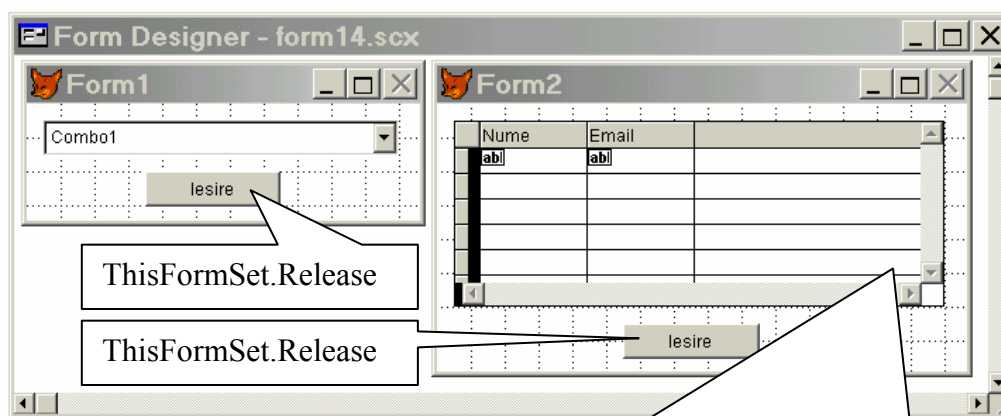
Să se realizeze o aplicație care să folosească două formulare care să conțină informațiile din baza de date *universitati*.

Rezolvare. Se urmează pașii:

1. Din *Project Manager* se selectează *Forms/New.../New Form*;
2. Din meniu, în categoria *Form* se selectează opțiunea *Create Form Set*;
3. Pe aceeași cale, se adaugă un nou formular: *Form/Add New Form*;

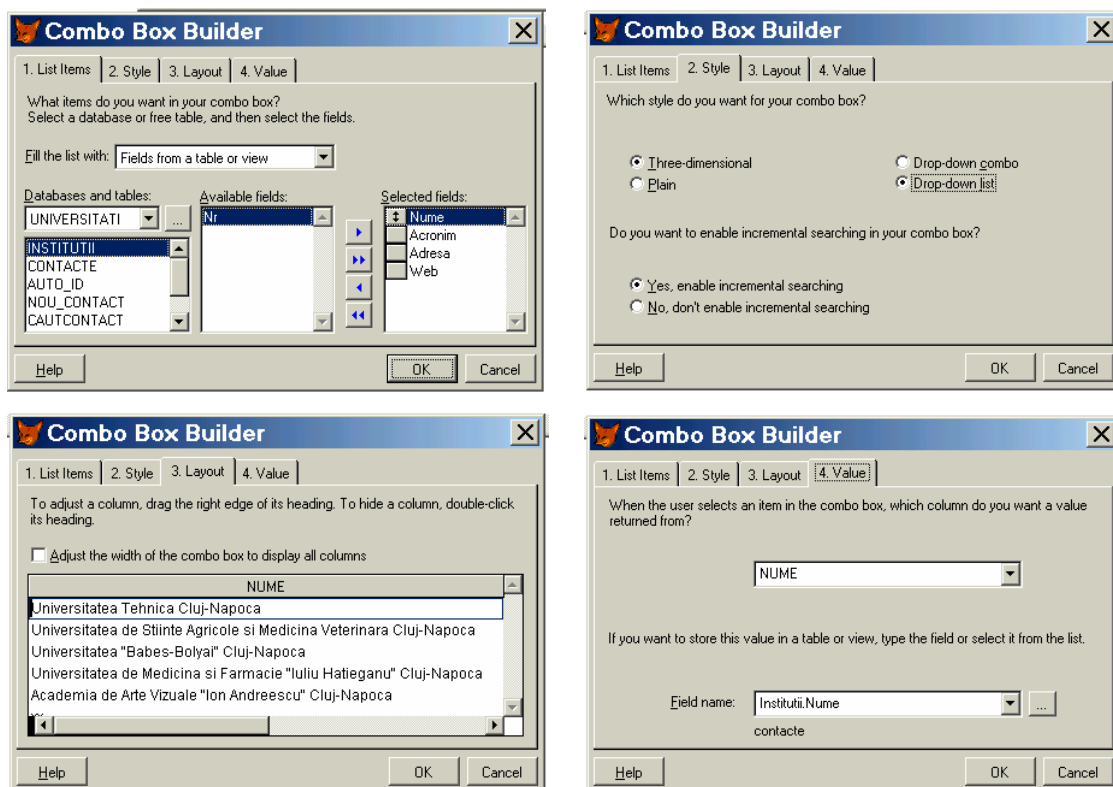


4. Se adaugă tabelele *institutii* și *contacte* în *Data Environment* (*View/Data Environment...*);
5. Se adaugă contoale și containere ca în figură:



6. Se folosește aplicația expert *Combo Box Builder* pentru a seta proprietățile casetei Combo1:

Crearea și exploatarea bazelor de date relaționale

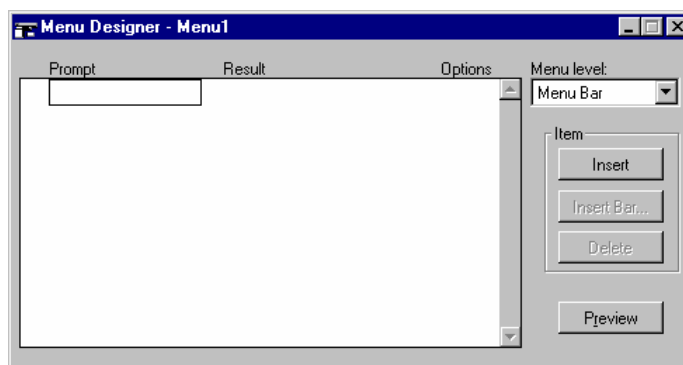


7. Se execută setul de formulare.

28. Meniuri

În VFP, folosirea meniurilor în aplicații face ca rezultatul obținut (produsul program) să își mărească calitatea în exploatare.

Fiecare parte a unei aplicații VFP poate avea propriul său meniu sistem sau un set de meniuri. Pentru a crea un meniu, se folosește aplicația expert *Menu Designer*.



Crearea unui meniu sistem implică câțiva pași. În funcție de mărimea aplicației și complexitatea meniurilor care se intenționează a se folosi, aceștia sunt:

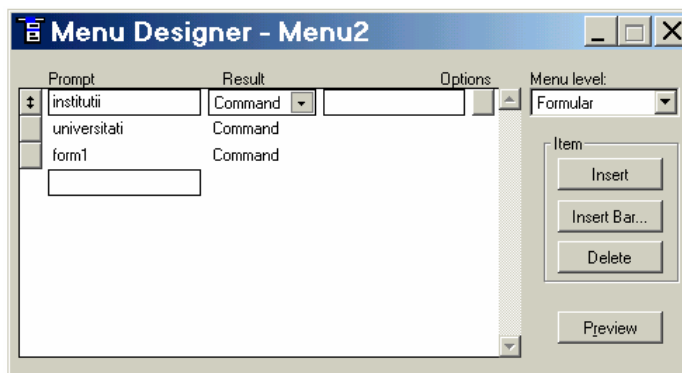
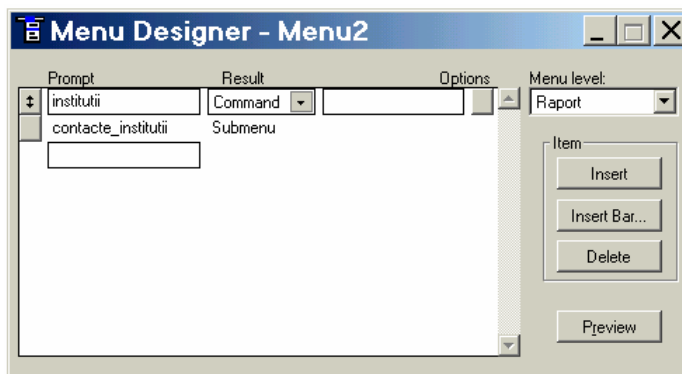
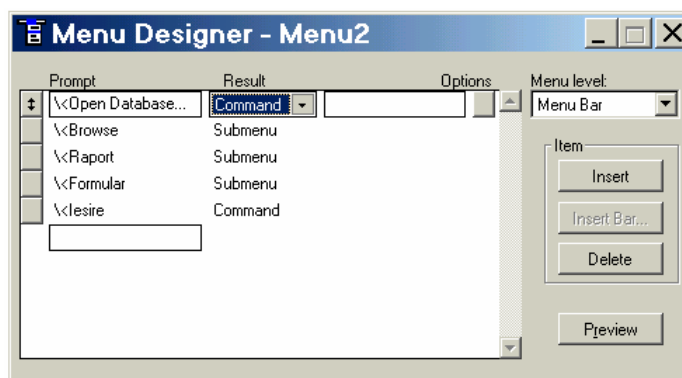
1. Planificarea și construcția sistemului (decide ce meniuri ai nevoie, dacă acestea vor apărea în interfață, care dintre acestea necesită submeniuri, și așa mai departe);
2. Crearea meniurilor și submeniurilor (definirea titlurilor pentru meniu, pentru elementele din meniu, utilizând *Menu Designer*);

Crearea și exploatarea bazelor de date relaționale

3. Asocierea de evenimente astfel încât sistemul să facă ce dorim (specificarea acțiunilor pentru meniuri de efectuat; adițional, se poate include cod de inițializare și cod de curățire); codul de inițializare se execută înainte ca meniul sistem să apară și se poate include cod pentru deschiderea de fișiere, declararea de variabile; codul de curățire conține ceea ce se va executa după codul de definire a meniului și poate face meniul sau elementele din meniu selectabile sau neselectabile;
4. Generarea programului pentru meniu;
5. Execuția programului pentru a testa sistemul.

Aplicația 5.

Să se creeze un meniul după modelul:



29. Dezvoltarea de meniuri pentru aplicații

Crearea unui meniu pentru o aplicație este asociată de obicei cu integrarea tuturor componentelor aplicației într-un ansamblu care să permită execuția fiecărei componente. Finalitatea acestui proces reprezintă sistemul de gestiune al bazei (sau bazelor) de date client.

Pentru construcția meniului sistemului se poate folosi aplicația expert *Menu Builder* sau se poate construi direct prin instrucțiuni, așa cum se va vedea în codul generat de VFP în urma generării unui meniu.

Aplicația 1.

Să se construiască un meniu cu opțiuni de comandă pentru acțiunile: deschidere baza de date *Universitati*, afișare ferestre de Browse pentru tabelele *Institutii* și *Contacte* și o opțiune Help About.

Rezolvare. Se urmează pașii:

1. Se creează un director pentru stocarea fișierelor aplicației;
2. Se creează un nou proiect (*New/Project/New file*); fie acesta *proj1.pjx*;
3. În fereastra de comenzi se testează funcția *GetDir()*, *Sys(5)* și *Sys(2003)*:
GETDIR("", "default for the application") respectiv *Sys(5)* și *Sys(2003)*;
4. În *Project Manager* la categoria *Code* se alege *Programs* și aici *New...*;
5. În fereastra pe care sistemul *VFP* o deschide se va crea un nou program care va gestiona proiectul; fie acesta *Program1*;
6. Se introduc următoarele comenzi:

```
cale_veche = sys(5)+sys(2003)
messagebox("cale veche: "+cale_veche)
cale = GETDIR("", "default for the application")
if like(cale, "")=.F.
    set default to &cale
    messagebox("noua cale: "+cale)
endif
* comenzi pentru executia aplicatiei
* ...
* sfarsit aplicatie
```

7. Se salvează fișierul; ne asigurăm că acesta a fost salvat în directorul proiectului *proj1.pjx*;
8. În *Project Manager* se selectează programul *program1* și apoi din meniul VFP la *Project* se verifică că programul *program1* are opțiunea *Set Main* selectată (✓ *Set Main*); în caz contrar se selectează; se execută și testează programul (*Project Manager/Code/Programs/Program1/Run*);
9. Se creează un nou meniu sistem (*Project Manager/Other/Menus/New.../Menu*) cu ajutorul aplicației expert *Menu Designer*; acesta se poate realiza după structura:

|<File (Submenu)
 |<Open... (Procedure)
 |<Quit (Procedure)
|<Browse (Command) Browse
|<Help (Submenu)
 |<About... (Procedure)



10. Cu *Menu Designer* – *menu1.mnx* deschis se selectează din meniul *VFP Menu/Generate...* pentru a genera codul programului de meniu; acesta conține definițiile pentru comenzi (*BAR*) și submeniuri (*PAD/POPUP*); dintre fișierele generate de VFP sunt esențiale pentru sistem: *menu1.MNX* (care este o tabelă cu informații despre structura și conținutul meniului), *menu1.MNT* (un fișier memo) și *menu1.mpr* (care conține comenzile generate de aplicația expert pentru crearea meniului); în acest din urmă fișier se pot vedea comenzile generate (se poate deschide cu *Notepad*); de menționat că sistemul nu permite modificarea directă a acestui fișier, el fiind actualizat la execuție în conformitate cu definițiile din fișierul *MNX*;
11. În orice moment al construcției sale, se poate vedea configurația meniului (*Menu Designer* – *menu1.mnx/Preview*);
12. Execuția meniului (*Project Manager/Other/Menus/Menu1/Run*) este sortită eșecului; sistemul lansează meniul în execuție însă nu este pregătit pentru gestiunea acestuia;
13. Se aplică următoarele modificări meniului (din *Menu Designer* – *menu1*):
- Se editează conținutul procedurii pentru *Help/About*:

```
MessageBox("First menu VFP");
```

- Se editează conținutul procedurii pentru *File/Quit*:

```
MessageBox("Now exit the application")
set default to cale_veche+"\"
MessageBox("restaurare cale: "+sys(5)+sys(2003))
Cancel
```

- Din meniul VFP se selectează *View/General Options...* și apoi din containerul de butoane de opțiune *Location* se selectează opțiunea *Append*;
 - Se salvează meniul (*File/Save*) și se generează noul cod (*Menu/Generate...*);
14. Se aplică următoarele modificări programului *program1*:

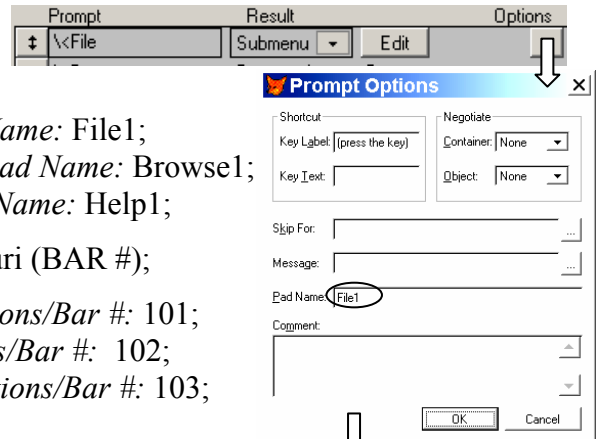
```
* comenzi pentru executia aplicatiei
do menu1.mpr
read events
* sfarsit aplicatie
```

15. Tentativa de a executa meniul este în continuare sortită eșecului; se poate însă acum executa *program1* (*Project Manager/Code/Programs/Program1/Run*);
16. Programul produce afișarea casetelor de mesaj dorite (*Help/About...* și *File/Quit*); observație: meniul aplicației se adaugă meniului sistemului *VFP* și la ieșire rămâne activat

Crearea și exploatarea bazelor de date relaționale

devenind parte integrantă a mediului VFP și după terminarea aplicației (comanda *Cancel*); pentru a ajusta această situație trebuie scrise comenzile corespunzătoare în procedura pentru comanda *File/Quit*; un alt inconvenient este starea activă a comenzii (opțiunii) *Browse* chiar dacă nu a fost selectată nici o tabelă (*File/Open...*);

17. Se modifică meniul pentru a asocia nume *PAD*-urilor (*Menu Designer – menu1*); se ține seama de faptul că numele implicite ale meniurilor VFP sunt în forma: (Edit: nume PAD: *_MSM_EDIT*) și nu se denumesc la fel!



18. Astfel, se pot denumi pe calea:

- \<File (Submenu) Options/Pad Name: File1;
- \<Browse (Command) Options/Pad Name: Browse1;
- \<Help (Submenu) Options/Pad Name: Help1;

19. Se pot asocia (însă numere) și pentru BAR-uri (BAR #);

- \<Open... (Procedure) Options/Bar #: 101;
- \<Quit (Procedure) Options/Bar #: 102;
- \<About... (Procedure) Options/Bar #: 103;

20. Se generează codul (*Menu/Generate...*);

21. Se vizualizează codul generat pentru a ne asigura de modificări;

22. Pentru dezactivarea lui *Browse* din meniu în *program1*:

do menu1.mpr

Set Skip of Pad Browse1 of _MSYSMENU.T.

read events

23. Se execută programul *program1*;

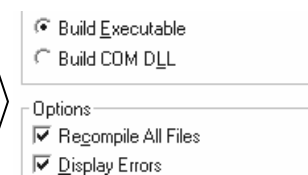
24. Pentru revenirea la meniul normal este acum necesară reîncărcarea aplicației *MVFP*;

25. Pentru ca aplicația să restaureze mediul VFP implicit se va modifica sfârșitul procedurii corespunzătoare comenzii *File/Quit* astfel:

messagebox("restaurare cale: "+sys(5)+sys(2003))

SET SYSMENU TO DEFAULT

cancel



26. Se poate genera acum aplicația executabilă pe calea *Project Manager – proj1/Code/Programs/program1/Build...*;

27. Se testează aplicația (din *Windows Explorer*); se completează acum cu procedurile lipsă;

28. La începutul programului *program1* se va adăuga o variabilă *tabela_use*:

tabela_use = ""

cale_veche = sys(5)+sys(2003)

29. Se încarcă din nou *menu1* în *Menu Designer*; se completează procedura pentru

File/Open...: **Menu Designer - menu1 - Open Procedure**

```
tabela_use1 = GetFile('DBF', 'Browse a Table:', 'Browse', 0, 'Browse')
if like(tabela_use, tabela_use1)
    Return
    && do nothing
```

Crearea și exploatarea bazelor de date relaționale

```
endif
if like(tabela_use1, "")
    Return                                && do nothing
endif
if Used(tabela_use1)
    use                                    && inchide tabela precedenta
    i = 1
    On Error i = 0
        USE &tabela_use1 in 0 again shared
    On Error
    if (i = 0)
        use &tabela_use                && redeschide tabela precedenta
        MsgBox("Table opened in exclusive mode by another program")
        Return
    else
        tabela_use = tabela_use1
        Set Skip of Pad Browse1 of _MSYSMENU .F.
        return
    endif
endif
use                                    && inchide tabela precedenta
i = 1
On Error i = 0
    USE &tabela_use1
On Error
if (i = 0)
    use &tabela_use                && redeschide tabela precedenta
    MsgBox("Table opened in exclusive mode by another program")
    Return
else
    tabela_use = tabela_use1
    Set Skip of Pad Browse1 of _MSYSMENU .F.
endif
```

30. Se poate adăuga un BAR Close (File/Close) a cărei procedură să dezactiveze pe Browse:

```
use
tabela_use = ""
Set Skip of Pad Browse1 of _MSYSMENU .T.
```

31. Se lansează în execuție și se testează aplicația *program1*; se generează executabilul;

32. Se pot elibera din memorie inclusiv elementele sistem, ca în exemplul:

RELEASE PAD _MEDIT OF _MSYSMENU

Aplicația 2.

Un meniu creat fără constructorul de meniuri.

Soluție. Se creează un nou program (File/New/Program/New file); se salvează cu numele Definpad.prg.

```
filen = Locfile("definpad", "prg.app.exe", "Locate definpad.prg")
set default to substr(filen,1,RAT("DEFINPAD.",filen)-1)
Clear
```

Crearea și exploatarea bazelor de date relaționale

```
Set Talk Off
Set Sysmenu Save
Set Sysmenu To
Public Markpad
Markpad = .T.
Define Pad Syspad Of _Msysmenu Prompt '\<System' Color Scheme 3 ;
    Key Alt+S, "
Define Pad Editpad Of _Msysmenu Prompt '\<Edit' Color Scheme 3 ;
    Key Alt+E, "
Define Pad Recordpad Of _Msysmenu Prompt '\<Record' Color Scheme 3 Key Alt+R, "
Define Pad Windowpad Of _Msysmenu Prompt '\<Window' Color Scheme 3 ;
    Key Alt+W, "
Define Pad Reportpad Of _Msysmenu Prompt 'Re\<Ports' Color Scheme 3 ;
    Key Alt+P, "
Define Pad Exitpad Of _Msysmenu Prompt 'E\<Xit' Color Scheme 3 ;
    Key Alt+X, "
On Selection Menu _Msysmenu ;
    Do Choice In Def̃inpad With Pad( ), Menu( )
Procedure Choice
Parameter Mpad, Mmenu
Wait Window 'You Chose ' + Mpad + ;
    ' From Menu ' + Mmenu Nowait
Set Mark Of Pad (Mpad) Of _Msysmenu To ;
    ! Mrkpad(' _Msysmenu', Mpad)
Markpad = ! Markpad
If Mpad = 'EXITPAD'
    Set Sysmenu To Default
Endif
```

Aplicația 3.

Un meniu creat fără constructorul de meniuri.

Soluție. Se creează un nou program (File/New/Program/New file); se salvează cu numele Def̃inbar.prg.

```
filen = Locfile("def̃inbar", "prg.app.exe", "Locate def̃inbar.prg")
set default to substr(filen,1,RAT("DEFINBAR."filen)-1)
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
DEFINE PAD convpad OF _MSYSMENU PROMPT '\<Conversions' COLOR SCHEME 3 ;
    KEY ALT+C, "
DEFINE PAD cardpad OF _MSYSMENU PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, "
ON PAD convpad OF _MSYSMENU ACTIVATE POPUP conversion
ON PAD cardpad OF _MSYSMENU ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
```

Crearea și exploatarea bazelor de date relaționale

```
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
  KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
  KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
  KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
  KEY CTRL+M, '^M'
ON SELECTION POPUP conversion;
  DO choice IN definbar WITH PROMPT(), POPUP()
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
  KEY ALT+V, "
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
  KEY ALT+P, "
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' KEY ALT+W, "
DEFINE BAR 4 OF cardinfo PROMPT '\-'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
DEFINE BAR 6 OF cardinfo PROMPT '\-'
DEFINE BAR 7 OF cardinfo PROMPT 'E\<xit '
ON SELECTION POPUP cardinfo;
  DO choice IN definbar WITH PROMPT(), POPUP()
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
  'from popup ' + mpopup NOWAIT
IF mprompt = 'Exit'
  SET SYSMENU TO DEFAULT
ENDIF
```

Aplicația 4.

Un meniu creat fără constructorul de meniuri.

Soluție. Se creează un nou program (File/New/Program/New file); se salvează cu numele Definmenu.prg.

```
filen = Locfile("definmenu", "prg,app,exe", "Locate definmenu.prg")
set default to substr(filen,1,RAT("DEFINMENU.",filen)-1)
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
ON KEY LABEL ESC KEYBOARD CHR(13)
DEFINE MENU example BAR AT LINE 1
DEFINE PAD convpad OF example PROMPT '\<Conversions' COLOR SCHEME 3 ;
  KEY ALT+C, "
DEFINE PAD cardpad OF example PROMPT 'Card \<Info' COLOR SCHEME 3 ;
  KEY ALT+I, "
ON PAD convpad OF example ACTIVATE POPUP conversion
ON PAD cardpad OF example ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' ;
```

Crearea și exploatarea bazelor de date relaționale

```
KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
KEY CTRL+M, '^M'
ON SELECTION POPUP conversion DO choice IN defimenu WITH PROMPT(), POPUP()
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
KEY ALT+V, "
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
KEY ALT+P, "
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' ;
KEY ALT+W, "
DEFINE BAR 4 OF cardinfo PROMPT '\<'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
ON SELECTION POPUP cardinfo;
DO choice IN defimenu WITH PROMPT(), POPUP()
ACTIVATE MENU example
DEACTIVATE MENU example
RELEASE MENU example EXTENDED
SET SYSMENU TO DEFAULT
ON KEY LABEL ESC
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
'from popup ' + mpopup NOWAIT
```

Aplicația 5.

Un meniu creat fără constructorul de meniuri cu submeniuri.

Soluție. Se creează un nou program (File/New/Program/New file). Se salvează și se execută.

```
DEFINE WINDOW wOrder FROM 10,0 TO 13,39
DEFINE MENU mnuDinner
DEFINE PAD padOne OF mnuDinner PROMPT '\<Main Course' KEY ALT+M, "
DEFINE PAD padTwo OF mnuDinner PROMPT '\<Dessert' KEY ALT+D, "
ON PAD padOne OF mnuDinner ACTIVATE POPUP popMainCourse
ON PAD padTwo OF mnuDinner ACTIVATE POPUP dessert
DEFINE POPUP popMainCourse MARGIN MESSAGE ;
'We have burgers and pizza today'
DEFINE BAR 1 OF popMainCourse PROMPT '\<Hamburgers'
DEFINE BAR 2 OF popMainCourse PROMPT '\<Pizza'
ON BAR 1 OF popMainCourse ACTIVATE POPUP burger
ON BAR 2 OF popMainCourse ACTIVATE POPUP pizza
```

Crearea și exploatarea bazelor de date relaționale

```
DEFINE POPUP burger MARGIN MESSAGE ;
  'What would you like on your burger?'
DEFINE BAR 1 OF burger PROMPT '\<Ketchup'
DEFINE BAR 2 OF burger PROMPT '\<Mustard'
DEFINE BAR 3 OF burger PROMPT '\<Onions'
DEFINE BAR 4 OF burger PROMPT '\<Pickles'
DEFINE POPUP pizza MARGIN MESSAGE 'Here are the available toppings'
DEFINE BAR 1 OF pizza PROMPT '\<Anchovies'
DEFINE BAR 2 OF pizza PROMPT '\<Green Peppers'
DEFINE BAR 3 OF pizza PROMPT '\<Olives'
DEFINE BAR 4 OF pizza PROMPT '\<Pepperoni'
ON BAR 3 OF pizza ACTIVATE POPUP olives
DEFINE POPUP olives MARGIN
DEFINE BAR 1 OF olives PROMPT '\<Black' MESSAGE 'Black olives?'
DEFINE BAR 2 OF olives PROMPT '\<Green' MESSAGE 'Green olives?'
DEFINE POPUP dessert MARGIN MESSAGE 'Our dessert offerings'
DEFINE BAR 1 OF dessert PROMPT '\<Brownies'
DEFINE BAR 2 OF dessert PROMPT '\<Cookies'
DEFINE BAR 3 OF dessert PROMPT '\<Ice Cream'
DEFINE BAR 4 OF dessert PROMPT '\<Pie'
ON BAR 4 OF dessert ACTIVATE POPUP pie
DEFINE POPUP pie MARGIN MESSAGE 'What kind of pie?'
DEFINE BAR 1 OF pie PROMPT '\<Blueberry'
DEFINE BAR 2 OF pie PROMPT '\<Cherry'
DEFINE BAR 3 OF pie PROMPT '\<Peach'
DEFINE BAR 4 OF pie PROMPT '\<Rhubarb'
ON SELECTION POPUP ALL DO yourchoice
ACTIVATE MENU mnuDinner
PROCEDURE yourchoice
ACTIVATE WINDOW wOrder
CLEAR
DO CASE
  CASE POPUP() = 'BURGER'
    @ 0,0 SAY 'A ' + POPUP() + ' order:'
    @ 1,0 SAY 'You ordered a burger with ' + LOWER(PROMPT())
  CASE POPUP() = 'PIZZA'
    @ 0,0 SAY 'A ' + POPUP() + ' order:'
    @ 1,0 SAY 'You ordered a pizza with ' + LOWER(PROMPT())
  CASE POPUP() = 'OLIVES'
    @ 0,0 SAY 'A ' + POPUP() + ' order:'
    @ 1,0 SAY 'You ordered a pizza with ' ;
      + LOWER(PROMPT()) + ' olives'
  CASE POPUP() = 'DESSERT'
    @ 0,0 SAY 'A ' + POPUP() + ' order:'
    @ 1,0 SAY 'You ordered ' + LOWER(PROMPT()) + ' for dessert'
  CASE POPUP() = 'PIE'
    @ 0,0 SAY 'A ' + POPUP() + ' order:'
    @ 1,0 SAY 'You ordered ' + LOWER(PROMPT()) + ' pie'
ENDCASE
WAIT WINDOW
DEACTIVATE WINDOW wOrder
RETURN
```

Aplicația 6.

Un meniu creat fără constructorul de meniuri.

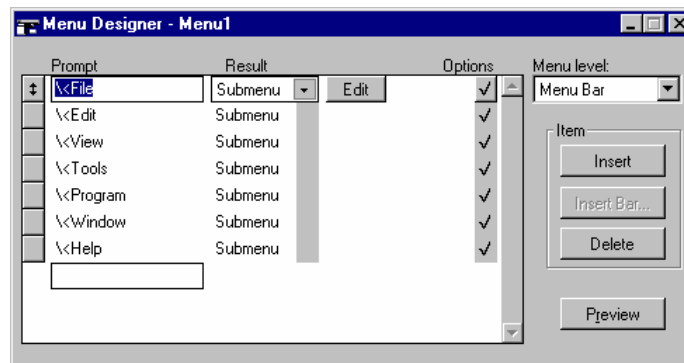
Soluție. Se creează un nou program (File/New/Program/New file); se salvează cu numele Definpop.prg.

```
filen = Locfile("definpop", "prg.app.exe", "Locate definpop.prg")
set default to substr(filen,1,RAT("DEFINPOP.",filen)-1)
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
DEFINE PAD conypad OF _MSYMENU PROMPT '\<Conversions' COLOR SCHEME 3 ;
    KEY ALT+C, "
DEFINE PAD cardpad OF _MSYMENU PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, "
ON PAD conypad OF _MSYMENU ACTIVATE POPUP conversion
ON PAD cardpad OF _MSYMENU ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
    KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
    KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
    KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
    KEY CTRL+M, '^M'
ON SELECTION POPUP conversion;
    DO choice IN definpop WITH PROMPT(), POPUP()
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
    KEY ALT+V, "
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
    KEY ALT+P, "
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' ;
    KEY ALT+W, "
DEFINE BAR 4 OF cardinfo PROMPT '\-'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
DEFINE BAR 6 OF cardinfo PROMPT '\-'
DEFINE BAR 7 OF cardinfo PROMPT 'E\<xit '
ON SELECTION POPUP cardinfo;
    DO choice IN definpop WITH PROMPT(), POPUP()
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ' from popup ' + mpopup NOWAIT
IF mprompt = 'Exit'
    SET SYSMENU TO DEFAULT
ENDIF
```

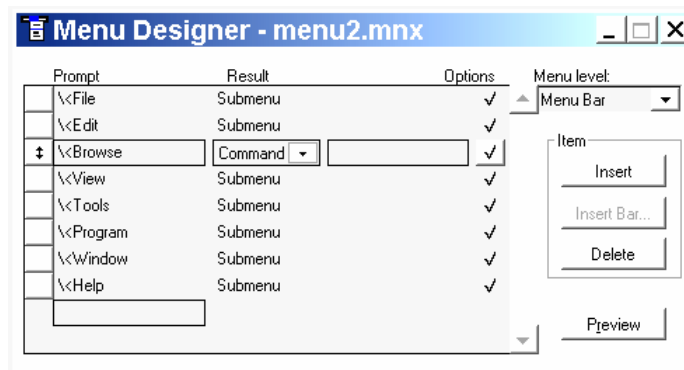
Generarea meniurilor cu *Quick Menu*

Sistemul VFP pune la dispoziția utilizatorului posibilitatea de a genera meniuri care să includă opțiunile de bază din mediul VFP. Este avantajos să se folosească această facilități pentru a putea beneficia de resursele sistemului *Windows* (de exemplu de *Clipboard* prin intermediul meniului *Edit*). Pentru a folosi această resursă se poate merge pe calea:

1. Se creează un nou meniu în cadrul proiectului *proj1.pjx*; fie acesta *menu2.mnx* (vezi Aplicația 1); se salvează cu numele *menu2.mnx*;
2. Cu aplicația expert *Menu Designer – Menu1* activă se selectează *Quick Menu* din *Menu*;
3. Sistemul VFP va genera un meniu în forma:



4. Se poate acum personaliza acest meniu prin adăugarea de opțiuni și asocierea de evenimente; de exemplu se poate adăuga la acest meniu opțiunea *Browse* și modifica opțiunile *File* și *Help* și asocia la acestea evenimente la fel ca în *Aplicația 1* când meniul va arăta în forma:



5. Se generează meniul (din meniul VFP *Menu/Generate...*);
6. Se modifică programul *Program1* din proiectul *proj1* pentru a încărca acum meniul *menu2.mpr*:

```
tabela_use = ""
cale_veche = sys(5)+sys(2003)
messagebox("cale veche: "+cale_veche)
cale = GETDIR("", "default for the application")
if like(cale, "")=.F.
    set default to &cale
    messagebox("noua cale: "+cale)
```

Crearea și exploatarea bazelor de date relaționale

endif

** comenzi pentru executia aplicatiei*

do menu2.mpr

Set Skip of Pad Browse1 of _MSYSMENU .T.

read events

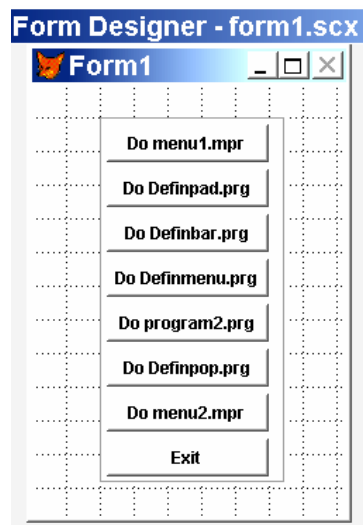
** sfarsit aplicatie*

7. Se execută programul *program1* (Project Manager – *proj1/Code/Programs/Program1/Run*); se generează executabilul (Project Manager – *proj1/Code/Programs/Program1/Build...*);

Execuția meniurilor din formulare

Odată create meniurile și generat codul sursă al programului de meniu se poate folosi acesta pentru a executa meniul din formulare. Următoarea aplicație folosește codul programelor *menu1.mpr*, *Definpad.prg*, *Definbar.prg*, *Definmenu.prg*, *program2.prg* (Aplicația 5), *Definpop.prg* și *menu2.mpr*. Se urmează pașii:

1. Se deschide proiectul *proj1.pjx* (Aplicația 1);
2. Se creează un nou formular (Project Manager – *proj1/Documents/Forms/New.../New Form*); se salvează cu numele *form1.scx*; se adaugă un container cu 8 butoane de comandă ca în figura:



3. Se asociază evenimente:

- a. Command1.Click:

Do menu1.mpr

Set Skip of Pad Browse1 of _MSYSMENU .T.

read events

- b. Command2.Click: *Do Definepad.prg*
- c. Command3.Click: *Do Definebar.prg*
- d. Command4.Click: *Do Definmenu.prg*
- e. Command5.Click: *Do program2.prg*

Crearea și exploatarea bazelor de date relaționale

- f. Command6.Click: *Do Definepop.prg*
- g. Command7.Click: *Do menu2.mpr*

Do menu2.mpr

*Set Skip of Pad Browse1 of _MSYSMENU .T.
read events*

- h. Command8.Click:

*SET SYSMENU TO DEFAULT
ThisForm.Release
Cancel*

- 4. Se modifică programul *program1*:

```
tabela_use = ""  
public cale_veche  
cale_veche = sys(5)+sys(2003)  
...  
* comenzi pentru executia aplicatiei  
Do Form form1.scx  
Read events  
set default to cale_veche+"\"  
messagebox("restaurare cale: "+sys(5)+sys(2003))  
* sfarsit aplicatie
```

- 5. Se modifică procedurile Menu1/Quit și Menu2/Quit și se regenerează codul meniurilor:

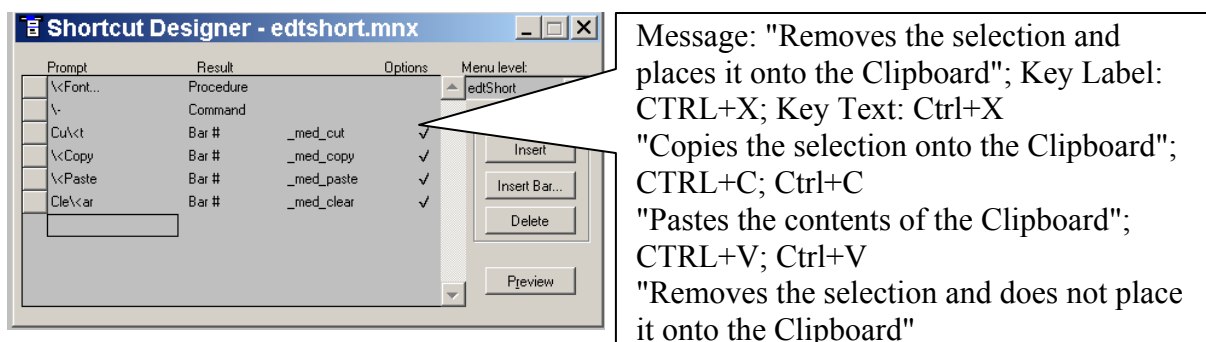
MessageBox("Now exit the application")
SET SYSMENU TO DEFAULT
Return to Master

- 6. Se execută formularul; se generează și testează executabilul.

Generarea de meniuri contextuale

Un meniu contextual se activează atunci când se execută click dreapta pe un control sau pe un obiect și reprezintă o modalitate rapidă de a afișa toate funcțiile aplicabile celui obiect. Un meniu contextual se realizează pe calea *Project Manager/Other/Menus/New...*

Următorul exemplu implementează un meniu contextual pentru operații de editare. Poate fi preluat direct pe calea MSDN98\98Vsa\1033\Samples\Solution\Vfp98\Solution când se copiază fișierele *Edtshort.mnx* și *Edtshort.MNT* după care se adaugă la proiect (*Project Manager – proj1/Other/Menus/Add.../Edtshort.mnx*).



Shortcut Designer - edtshort - Font... Procedure

```

IF TYPE("m.oRef") = "O"
    m.cFont = GetFont()
    IF EMPTY(m.cFont)
        RETURN
    ENDIF
    m.commaLoc = AT(", ", m.cFont)
    m.comma2Loc = AT(", ", m.cFont, 2)
    oRef.FontName = SUBSTR(m.cFont, 1, m.commaLoc-1)
    oRef.FontSize = VAL(SUBSTR(m.cFont, m.commaLoc+1, m.comma2Loc-m.commaLoc))
    oRef.FontBold = ATC("B", SUBSTR(m.cFont, m.comma2Loc))#0
    oRef.FontItalic = ATC("I", SUBSTR(m.cFont, m.comma2Loc))#0
ENDIF

```

7. Se generează codul meniului;
8. Se modifică formularul și se adaugă evenimentul:

FORM1.RightClick

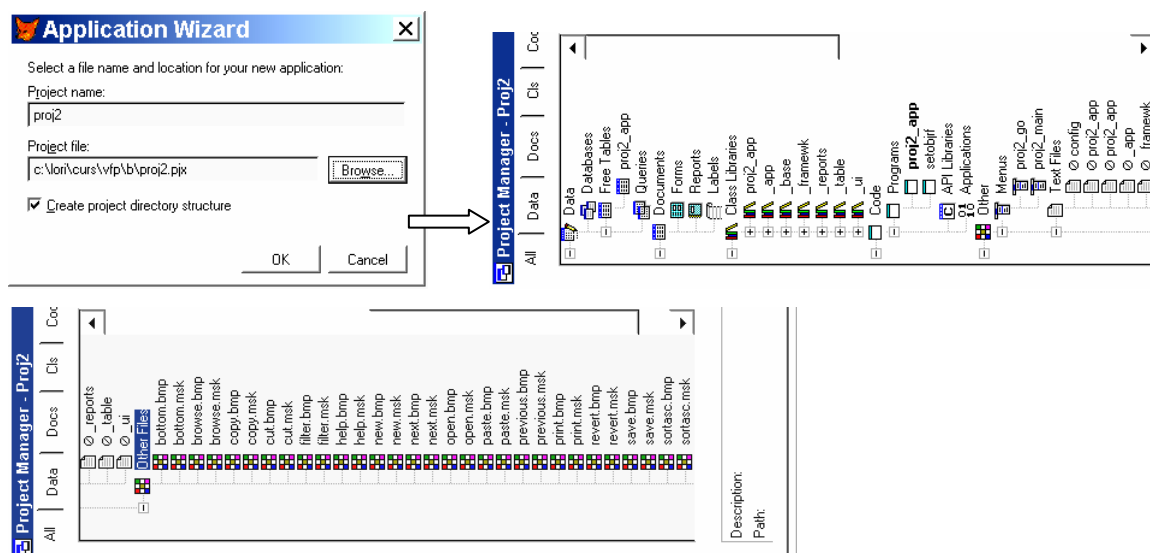
```
do Edtshort.mpr
read events
```

9. Se generează codul executabil și se testează aplicația *proj1*.

Generarea automată a aplicației cu aplicația expert *Application Wizard*

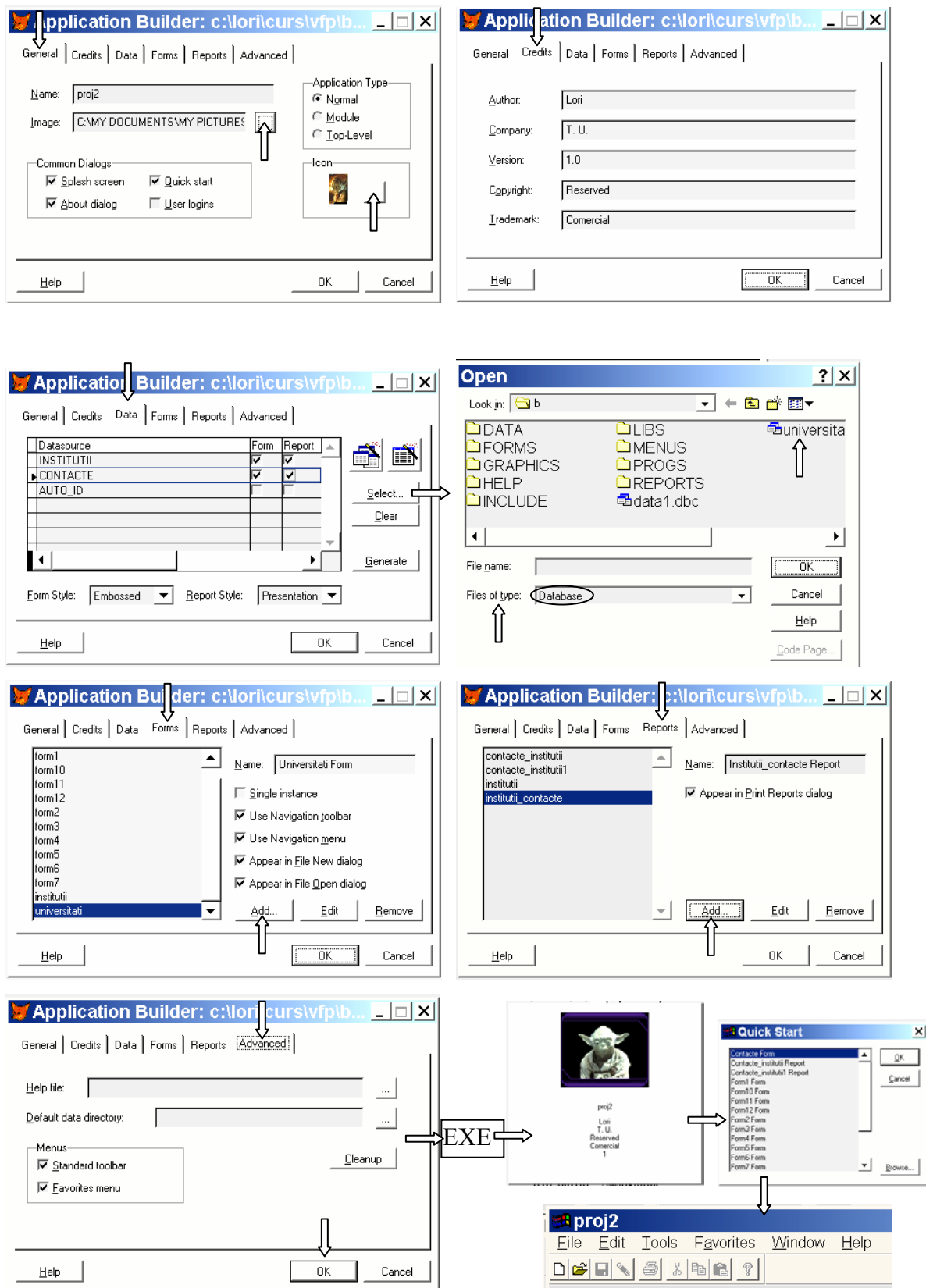
Se poate genera automat proiectul unei aplicații pe calea *File/New/Project/Wizard*. Următorul exemplu generează automat proiectul pentru baza de date *Universitati*. Se urmează pașii:

1. Se creează un nou director;
2. Se copiază fișierele din directorul proiectului Universitati mai puțin Universitati.PJT și Universitati.pjx;
3. Se lansează în execuție aplicația expert *Application Wizard*;
4. Se definesc atributele proiectului:



Crearea și exploatarea bazelor de date relaționale

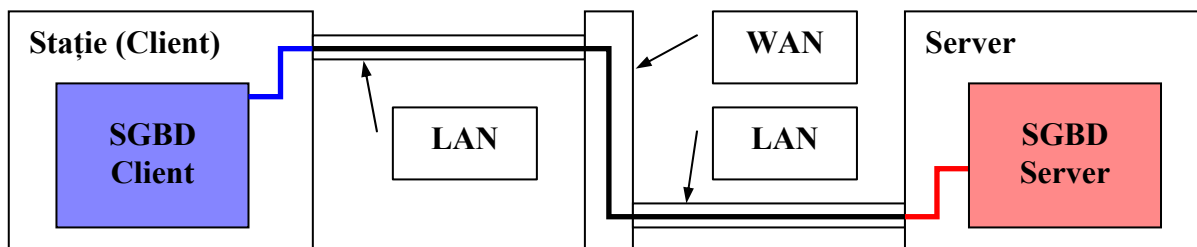
5. Sistemul VFP va genera un proiect GENERIC ca în figurile anterioare; în continuare cu ajutorul aplicației expert *Application Builder* se pot defini elementele constitutive ale proiectului:



Noul proiect generat include acum formularele și rapoartele create; se generează executabilul și se testează aplicația.

30. Baze de date externe și aplicații client – server

Aproape toate SGBD-urile care se execută sub sisteme de operare cu nucleu (kernel) client-server (Windows 3.11 NT, 9.x, NT 4.x, XP; Novell Netware 3.x, 4.x, 5.x, Linux, Unix, OS/2, BSD) au prevăzute protocoale de comunicare pentru baze de date situate la distanță (pe stații sau pe servere). Există o mare diversitate de sisteme de operare atât pentru stații (clienți) cât și pentru servere. Din acest motiv, comunicarea se realizează prin intermediul unui protocol, care reprezintă de fapt implementarea unor convenții în ceea ce privește comunicarea de date sub formă de librării și pachete de programe.



Protocoalele de transfer de date (și driverele aferente) ale stației și serverului sunt furnizate de către sistemul de operare, așa cum se poate observa în figurile de mai jos. Acestea se configurează urmând prescripțiile din documentația sistemului de operare și ale distribuitorului de internet (pentru WAN).

The following network components are installed:

- Client for Microsoft Networks
- Intel 21041 Based PCI Ethernet Adapter
- IPX/SPX-compatible Protocol
- NETBIOS support for IPX/SPX-compatible Protocol
- TCP/IP
- File and printer sharing for Microsoft Networks

Win 9.x

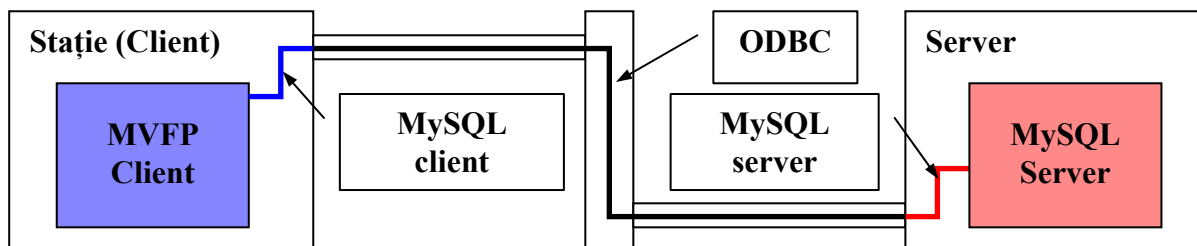
Free BSD

```

bash-2.05a$ ifconfig
de0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 193.226.7.140 netmask 0xfffff80 broadcast 193.226.7.255
    inet6 fe80::200:c0ff:fe91:ebf5%de0 prefixlen 64 scopeid 0x1
    ether 00:00:c0:91:eb:f5
    media: Ethernet autoselect (10baseT/UTP)
    status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
    inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 552
faith0: flags=8002<BROADCAST,MULTICAST> mtu 1500
        
```

Cazul cel mai complex este atunci când sistemele de operare sunt diferite și SGBD-urile trebuie configurate pentru comunicare atât pe server cât și pe stație.

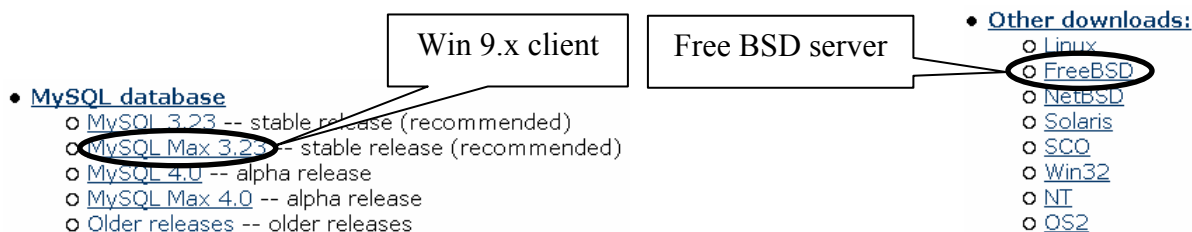
31. Configurarea unui client VFP/Win9.x pe un server MySQL/FreeBSD



Pentru a începe configurarea unui sistem de exploatare a bazelor de date la distanță trebuie să ne asigurăm că dispunem de toate driverele necesare. În cazul în care am ales să instalăm un Server MySQL va trebui să aducem componentele necesare și să le instalăm atât pe server cât și pe stații.

Se parcurg următorii pași:

1. Se alege pachetul de drivere și programe corespunzătoare sistemului de operare (www.mysql.com/downloads):



2. Se aduc și se instalează acestea; MySQL server va descărca și instala și alte pachete adiționale pentru limbajul perl :

post-patch:

```
{PERL} -pi -e \
"s@^(my $$Libs.*)-lpthread -ldl";$$@1 ${PTHREAD_LIBS}\";@; \
s@^(CFLAGS_GLOB.*)@1 ${CFLAGS} ${PTHREAD_CFLAGS}@" \
${WRKSRC}/configure.pl
```

3. Dacă dorim să asigurăm o interfață grafică SGBD-ului de pe server, se poate instala acum phpMyAdmin (<http://www.phpwizard.net/projects/phpMyAdmin/>) care este un sistem care administrează MySQL prin protocolul http. Acesta oferă interfața grafică pentru:
 - a. crearea și ștergerea bazelor de date;
 - b. crearea, copierea, ștergerea și modificarea tabelor;
 - c. ștergerea, editarea și adăugarea de câmpuri;
 - d. executarea oricărei secvențe în limbajul SQL și chiar a interogărilor în lanțuite;
 - e. manipularea cheilor în câmpuri;
 - f. încărcarea fișierelor text în tabele;
 - g. crearea și citirea de tabele;
 - h. exportul datelor;
 - i. administrarea multiplă de servere pe o singură bază de date.

Crearea și exploatarea bazelor de date relaționale

4. Cu ajutorul lui phpmyadmin (<http://www.lejpt.utcluj.ro/phpmyadmin>) se pot crea acum utilizatori care să acceseze bazele de date MySQL din interfața phpMyAdmin:

MySQL
Create new database [Documentation]
referenti [Create]
• Create new table on database referenti:
Name: lejpt
Fields: 5 [Go]
Field Type Length/Values*
nr TINYINT
nume TINYINT 30
tip TINYINT 15
functia TINYINT 20
domeniu TINYINT 20

• Add a new User
• Any host • Host :
• Any user • User name : read
• No Password • Password : Re-type :
Privileges :
☐ Select ☐ Insert
☐ Update ☐ Delete
☐ Create ☐ Drop
☒ Reload ☐ Shutdown
☒ Process ☐ File
☐ Grant ☐ References
☐ Index ☐ Alter
[Check All] [Uncheck All]
[Go]

Any host - User read

You have added a new user.
Remember reload the server.

SQL-query :
INSERT INTO mysql.user SET Host = '%', User = 'read', Password = PASSWORD('postuniv'), Select_priv = 'N', Insert_priv = 'N', Update_priv = 'N', Delete_priv = 'N', Create_priv = 'N', Drop_priv = 'N', Reload_priv = 'Y', Shutdown_priv = 'N', Process_priv = 'Y', File_priv = 'N', Grant_priv = 'N', References_priv = 'N', Index_priv = 'N', Alter_priv = 'N'

Note: MySQL privilege names are expressed in English

Action	Host	User	Password	Privileges
Edit Delete Grants	%	lori	Yes	Select Insert Update Delete Create Drop Reload Shutdown Process File Grant References Index Alter
Edit Delete Grants	%	read	Yes	Reload Process

5. Interogarea SQL este în forma următoare:

```
CREATE TABLE `lejpt` (`nr` TINYINT NOT NULL, `nume` TINYINT(30) NOT NULL, `tip` TINYINT(15) NOT NULL, `functia` TINYINT(20) NOT NULL, `domeniu` TINYINT(20) NOT NULL);
```

6. Se poate modifica acum structura tablei. De exemplu se corectează câmpurile nume, tip, functia, domeniu din tinyint în char și tipul indexului pentru câmpurile de tip caracter:

[Browse] [Select] [[Insert](#)] [Empty] [[Drop](#)]

	Field	Type	Attributes	Null	Default	Extra	Action
☐	nr	tinyint(4)		No	0		Change Drop Primary Index Unique Fulltext
☐	nume	tinyint(30)		No	0		Change Drop Primary Index Unique Fulltext
☐	tip	tinyint(15)		No	0		Change Drop Primary Index Unique Fulltext
☐	functia	tinyint(20)		No	0		Change Drop Primary Index Unique Fulltext
☐	domeniu	tinyint(20)		No	0		Change Drop Primary Index Unique Fulltext

With selected: [Change](#) Or [Drop](#)

Copy table to (database.table):
referenti . [Go]
☐ Structure only
☐ Structure and data

Type
tinyint(4)
char(30)
char(15)
char(20)
char(20)

Indexes : [Documentation]
Keyname Type Cardinality Action Field
PRIMARY PRIMARY 0 [Drop](#) [Edit](#) nr
nume INDEX None [Drop](#) [Edit](#) nume
tip INDEX None [Drop](#) [Edit](#) tip
functia INDEX None [Drop](#) [Edit](#) functia
domeniu INDEX None [Drop](#) [Edit](#) domeniu

Crearea și exploatarea bazelor de date relaționale

7. Se poate face o copie a structurii într-un nou tabel; fie acesta ljs; comanda SQL generată de phpMyadmin este:

```
CREATE TABLE `referenti`.`ljs` (`nr` tinyint(4) NOT NULL default '0', `nume` char(30) NOT NULL default '0', `tip` char(15) NOT NULL default '0', `functia` char(20) NOT NULL default '0', `domeniu` char(20) NOT NULL default '0', PRIMARY KEY (`nr`), KEY `nume` (`nume`), KEY `tip` (`tip`), KEY `functia` (`functia`), KEY `domeniu` (`domeniu`))
TYPE=MyISAM;
```

8. Acum baza de date referenti conține cele două tabele, lejpt și ljs;

9. Se poate defini câmpul nr ca câmp autoincrement:

```
ALTER TABLE `referenti`.`lejpt` CHANGE `nr` `nr`
TINYINT(4) DEFAULT '0' NOT NULL AUTO_INCREMENT
```

10. Se pot completa tabelele cu înregistrări pe calea:

[Browse] [Select] **[Insert]** [Empty] [Drop]

Insert another new row

Go

Value

1

Gheorghe LAZEA

referent

prof. dr.

automatics

referenti (2)

referenti

lejpt

ljs

11. Odată definită funcția autoincrement, nu mai este necesară completarea valorii pentru nr:

Value

Maria MICULA

referent

prof. dr.

mathematics

[Browse] [Select] [Insert] [Empty] [Drop]

	nr	nume	tip	functia	domeniu
Edit Delete	1	Gheorghe LAZEA	referent	prof. dr.	automatics
Edit Delete	2	Maria MICULA	referent	prof. dr.	mathematics

12. Comanda SQL este în forma:

```
INSERT INTO `lejpt` (`nr`, `nume`, `tip`, `functia`, `domeniu`) VALUES ('', 'Valentin MILITARU', 'referent', 'prof. dr.', 'fizica');
```

13. Corectarea unei valori introduse greșit se face cu comanda:

```
UPDATE `lejpt` SET `domeniu` = 'physics' WHERE `nr` = '3' LIMIT 1;
```

14. Rezultatul completării cu date poate arăta în felul următor:

	nr	nume	tip	functia	domeniu
Edit Delete	1	Gheorghe LAZEA	referent	prof. dr.	automatics
Edit Delete	2	Maria MICULA	referent	prof. dr.	mathematics
Edit Delete	3	Valentin MILITARU	referent	prof. dr.	physics
Edit Delete	4	Elena Maria PICA	referent	prof. dr.	chemistry
Edit Delete	5	Tiberiu RUSU	referent	prof. dr.	engineering
Edit Delete	6	Mircea SAVATTI	referent	prof. dr.	agronomy
Edit Delete	7	Lorentz JANTSCHI	chief redactor	sef lucr. dr.	informatics
Edit Delete	8	Florica ALDEA	redactor	sef lucr. dr.	applied mathematics
Edit Delete	9	LASZLO Alexandru	redactor	prof. dr. gr. 1	linguistics

referenti

lejpt

ljs

test

Drop table

Run SQL query/queries on datab

```
INSERT INTO `referenti`.`ljs` SELECT * FROM `referenti`.`lejpt`
```

Show this query here again

Or Location of the textfile :

Browse...

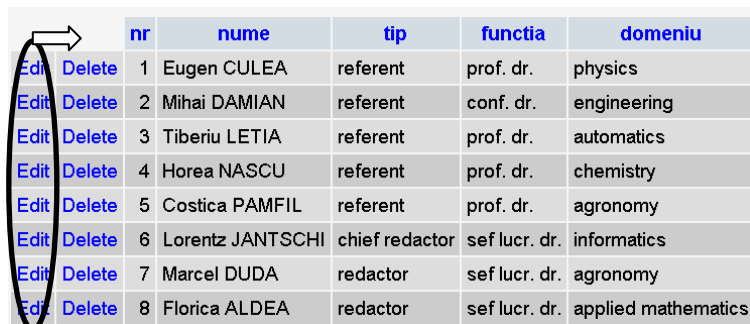
Go

15. Se pot acum copia valorile din tabelul lejpt în tabelul ljs:

```
INSERT INTO `referenti`.`ljs` SELECT * FROM `referenti`.`lejpt`
```

Crearea și exploatarea bazelor de date relaționale

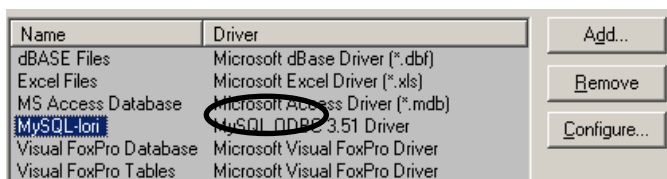
16. Dacă a fost creat un tabel care ulterior nu mai este de dorit a se păstra în baza de date se aplică comanda drop asupra acestuia;
17. După ce s-au copiat valorile din tabelul lejpt în tabelul ljs se pot modifica după dorință (întâi browse și apoi edit):



	nr	nume	tip	functia	domeniu
Edit Delete	1	Eugen CULEA	referent	prof. dr.	physics
Edit Delete	2	Mihai DAMIAN	referent	conf. dr.	engineering
Edit Delete	3	Tiberiu LETIA	referent	prof. dr.	automatics
Edit Delete	4	Horea NASCU	referent	prof. dr.	chemistry
Edit Delete	5	Costica PAMFIL	referent	prof. dr.	agronomy
Edit Delete	6	Lorentz JANTSCHI	chief redactor	sef lucr. dr.	informatics
Edit Delete	7	Marcel DUDA	redactor	sef lucr. dr.	agronomy
Edit Delete	8	Florica ALDEA	redactor	sef lucr. dr.	applied mathematics

Instalarea și administrarea MySQL pe client

Se poate alege a se instala MyODBC-3.51.01 care după instalare se va regăsi în Control Panel/ODBC Data Sources (32bit); se va selecta și se va configura pentru conectarea implicită pe serverul de date.



Pentru a permite accesul la citire în bazele de date de pe server, cu ajutorul lui phpMyAdmin se creează 3 utilizatori după modelul:

Edit Delete Grants %	read	Yes	Select Reload Process
Edit Delete Grants lejpt	read	Yes	Select Reload Process
Edit Delete Grants localhost	read	Yes	Select Reload Process

Username:

Password:

Login

Se testează accesul pentru utilizatorul nou creat:

MySQL 3.23.49 running on localhost as read@localhost

Tentativa de modificare a conținutului tabelelor este soldată eșecului:

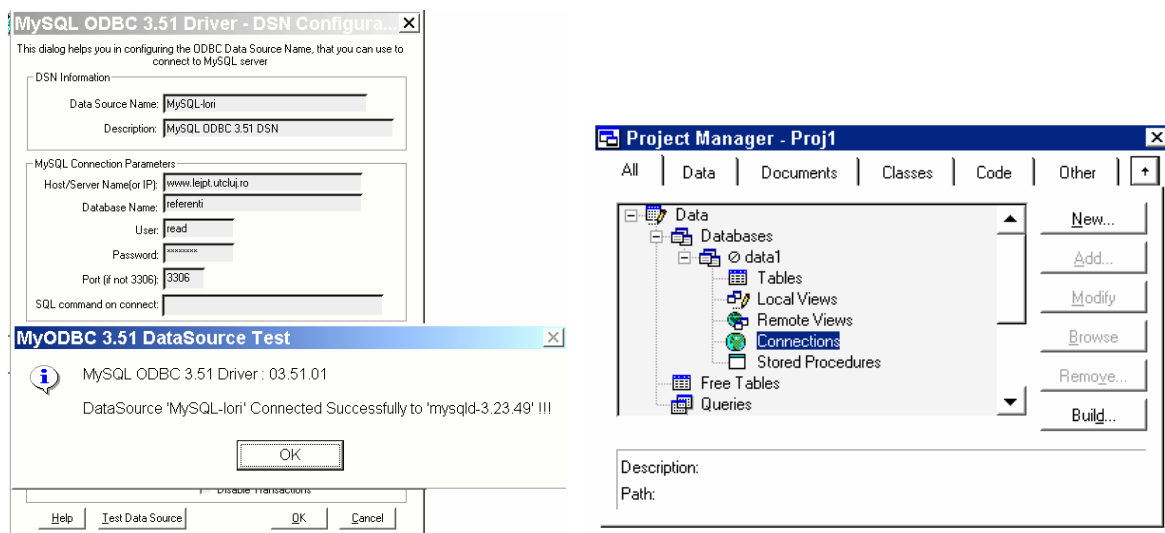
Edit Delete	1	Eugen CULEA	referent	prof. dr.	physics
-------------	---	-------------	----------	-----------	---------

DELETE FROM `ljs` WHERE `nr` = '1' LIMIT 1

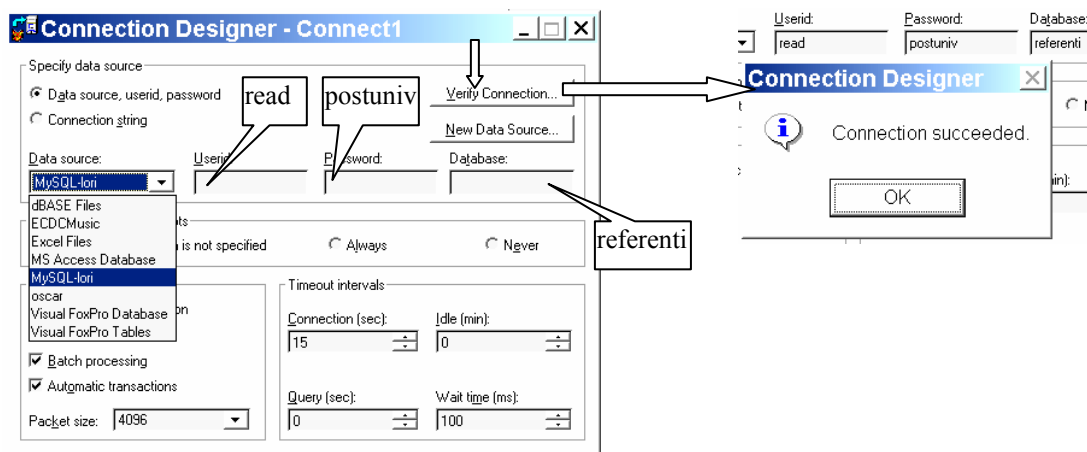
Access denied for user: 'read@localhost' to database 'referenti'

Se poate configura acum accesul de la distanță în baza de date, astfel încât în următoarea etapă se configurează MySQL pe client după modelul:

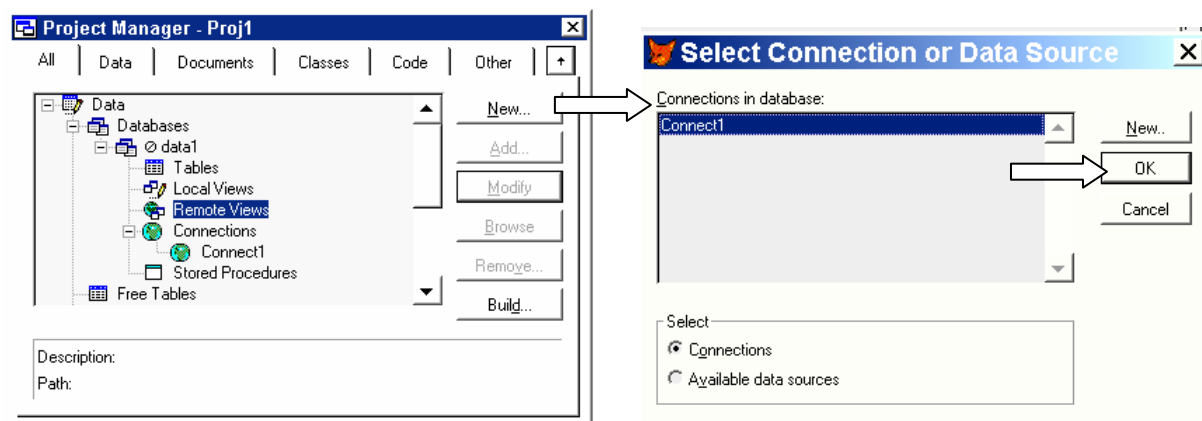
Crearea și exploatarea bazelor de date relaționale



Se poate acum accesa baza de date și din VFP. Se încarcă *Project Manager*, se deschide o bază de date sau se creează una nouă, se selectează *Connections*, apoi *New...* și se urmează schema de mai jos:

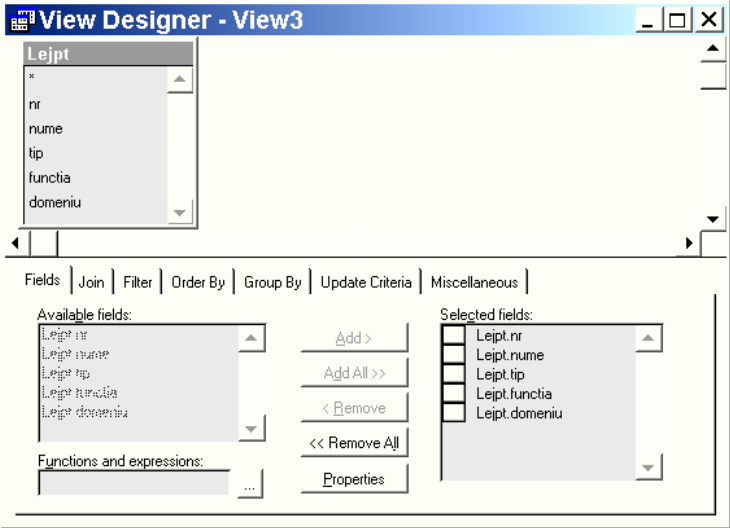


Se salvează conexiunea în baza de date (connect1) și se creează 2 vederi la distanță (Remote Views/New) din baza de date referenti:



Se poate crea o vedere care să conțină informațiile din tabelul ljs și o vedere care să conțină informațiile din tabelul lejpt. Se vor include toate câmpurile din cele două tabele și se vor salva acestea în baza de date:

Crearea și exploatarea bazelor de date relaționale



View Designer - View3

Tables in Connect1

- leipt
- ljs

Include System Tables ☐ All User Tables ☒

Available fields:

- Leipt.nr
- Leipt.num
- Leipt.tip
- Leipt.functia
- Leipt.domeniu

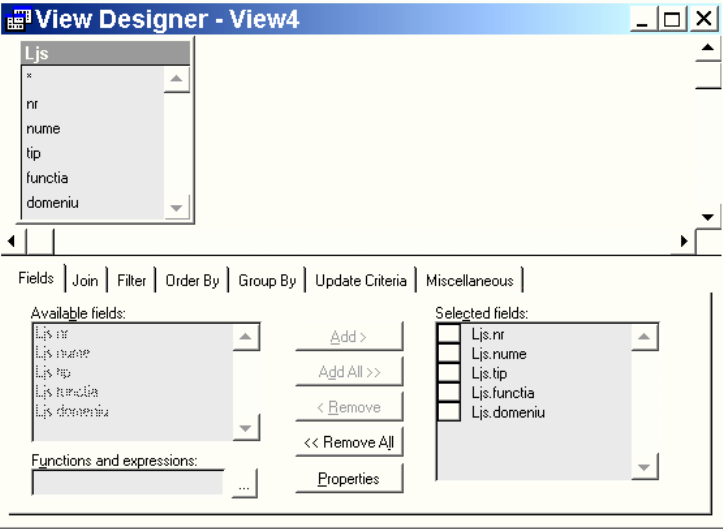
Selected fields:

- Leipt.nr
- Leipt.num
- Leipt.tip
- Leipt.functia
- Leipt.domeniu

Fields | Join | Filter | Order By | Group By | Update Criteria | Miscellaneous

Functions and expressions: ...

Properties



View Designer - View4

Tables in Connect1

- leipt
- ljs

Include System Tables ☐ All User Tables ☒

Available fields:

- Ljs.nr
- Ljs.num
- Ljs.tip
- Ljs.functia
- Ljs.domeniu

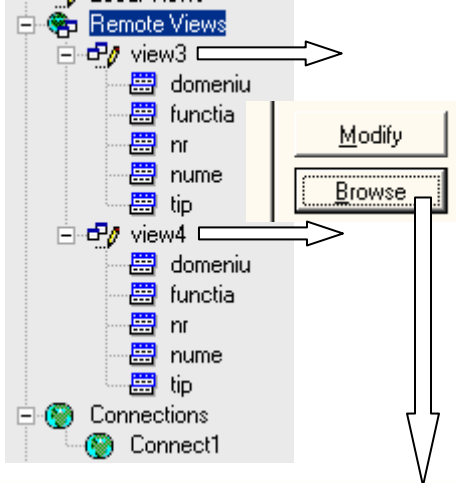
Selected fields:

- Ljs.nr
- Ljs.num
- Ljs.tip
- Ljs.functia
- Ljs.domeniu

Fields | Join | Filter | Order By | Group By | Update Criteria | Miscellaneous

Functions and expressions: ...

Properties



Remote Views

- view3
 - domeniu
 - functia
 - nr
 - nume
 - tip
- view4
 - domeniu
 - functia
 - nr
 - nume
 - tip
- Connections
 - Connect1

Modify

Browse

View3

Nr	Nume	Tip	Functia	Domeniu
1	Gheorghe LAZEA	referent	prof. dr.	automatics
2	Maria MICULA	referent	prof. dr.	mathematics
3	Valentin MILITARU	referent	prof. dr.	physics
4	Elena Maria PICA	referent	prof. dr.	chemistry
5	Tiberiu RUSU	referent	prof. dr.	engineering
6	Mircea SAVATTI	referent	prof. dr.	agronomy
7	Lorentz JANTSCHI	chief redactor	sef lucr. dr.	informatics
8	Florica ALDEA	redactor	sef lucr. dr.	applied mathematics
9	LASZLO Alexandru	redactor	prof. dr. gr. 1	linguistics

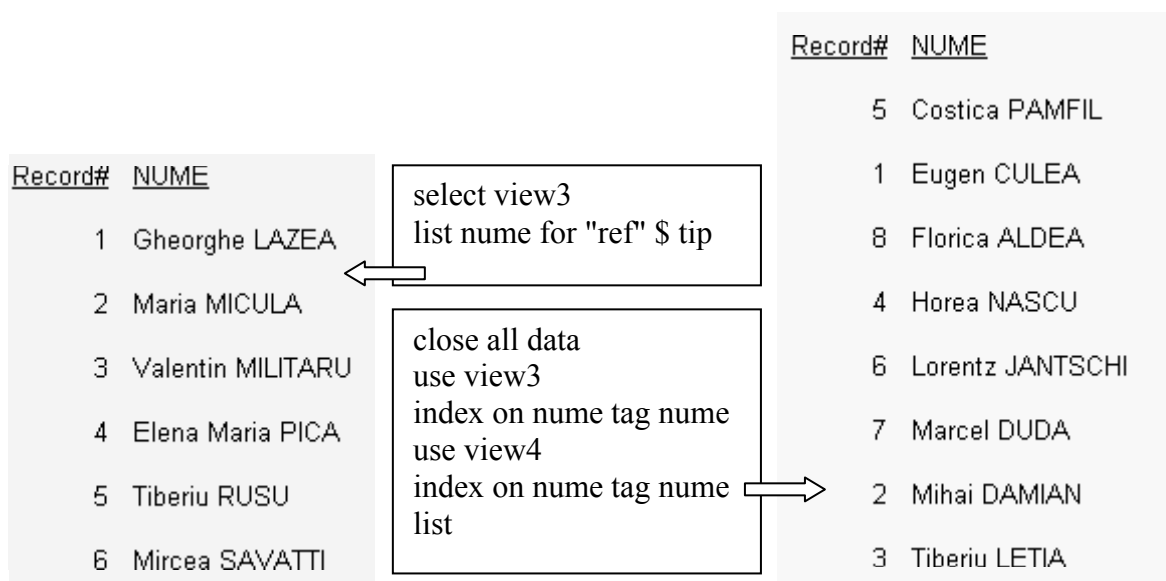
View4

Nr	Nume	Tip	Functia	Domeniu
1	Eugen CULEA	referent	prof. dr.	physics
2	Mihai DAMIAN	referent	conf. dr.	engineering
3	Tiberiu LETIA	referent	prof. dr.	automatics
4	Horea NASCU	referent	prof. dr.	chemistry
5	Costica PAMFIL	referent	prof. dr.	agronomy
6	Lorentz JANTSCHI	chief redactor	sef lucr. dr.	informatics
7	Marcel DUDA	redactor	sef lucr. dr.	agronomy
8	Florica ALDEA	redactor	sef lucr. dr.	applied mathematics

Se pot executa vederile pe calea Remote Views/View3(4)/Browse. De asemenea, se pot realiza interogări, rapoarte și formulare, acestea fiind tratate în continuare ca elemente aparținătoare bazei de date locale.

Crearea și exploatarea bazelor de date relaționale

Se pot da și comenzi în fereastra de comenzi sau crea programe care să folosească aceste informații:



```
SELECT VIEW4.Nume; FROM DATA1!VIEW4 VIEW4;  
INNER JOIN DATA1!VIEW3 VIEW3 ON VIEW4.NUME = VIEW3.NUME;  
ORDER BY VIEW4.Nume to screen
```

Se poate acum crea un executabil care să acceseze datele din conexiunea la distanță.

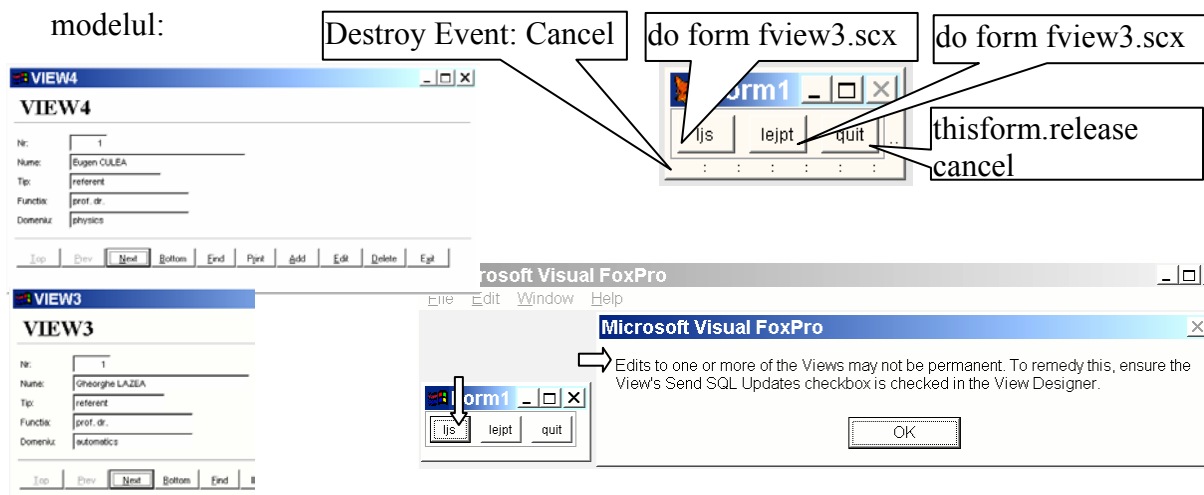
Se urmează pașii:

1. Se creează un program principal pentru aplicație (Project Manager – proj1/Code/Programs/New...); fie acesta program1.prg;
2. Se introduc în acesta comenzile:

set default to GETDIR("", "default for the application")

do form form1.scx

3. Se setează ca program principal al proiectului: Project Manager – proj1/Code/Programs/program1/Project/✓ Set Main;
4. Se creează formularul form1 pe calea Project Manager – proj1/Forms/New Form după modelul:



32. Administrarea de la distanță a MySQL/FreeBSD cu VFP/Win9.x

După instalarea MySQL pe server pentru a configura pentru acces partajat și proteja serverul se poate folosi programul *mysql_setpermission*. Execuția acestui program necesită privilegii de administrator (su-2.05a#). În continuare configurarea serverului cu acest program se face prin intermediul unei interfețe de tip text:

Password for user to connect to MySQL:

```
#####
## Welcome to the permission setter 1.2 for MySQL.
#####
```

What would you like to do:

1. Set password for a user.
2. Add a database + user privilege for that database.
 - user can do all except all admin functions
3. Add user privilege for an existing database.
 - user can do all except all admin functions
4. Add user privilege for an existing database.
 - user can do all except all admin functions + no create/drop
5. Add user privilege for an existing database.
 - user can do only selects (no update/delete/insert etc.)
0. exit this program

Make your choice [1,2,3,4,5,0]:

Prin această interfață se poate defini parola pentru superuser (root) și se pot defini parolele și drepturile de acces pentru toți utilizatorii serverului.

În continuare, administratorul poate configura și exploata programatic serverul MySQL din interfața MVFP.

Conectarea la server se poate face cu comanda:

```
nConnectionHandle = SQLCONNECT('sqlremote','sa','secret')
```

unde *sqlremote* este numele conexiunii (MySQL-lori), *sa* este numele superuserului (root) iar *secret* este parola acestuia (*****).

Pentru crearea unei interfețe grafice de administrare a serverului (MySQL-lori) în MVPF se pot urma pașii:

1. Se creează un nou proiect (New/Project/New File);
2. Se creează un nou formular (Project Manager – Proj1/Documents/Forms/New.../New Form); se construiește formularul:

Label1: Caption = Server connection

sqldisconnect(nCH)
release nCH (:form1.destroy)

Form1

Server connection name: Text1

User name: Text2

Password: Text3

Connect Cancel

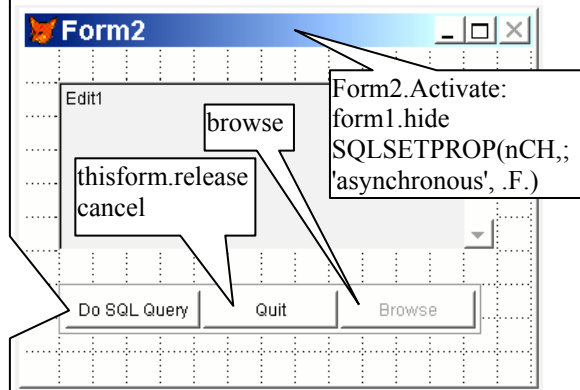
Command2.Click:
sqldisconnect(nCH)
release nCH
thisform.release
cancel

Text3.InteractiveChange: (la fel Text1 și Text2)
if ((len(alltrim(thisform.text1.text))>0) AND;
(len(alltrim(thisform.text2.text))>0) AND;
(len(alltrim(thisform.text3.text))>0))
thisform.commandgroup1.command1.enabled = .T.
endif
if ((len(alltrim(thisform.text1.text))=0) OR;
(len(alltrim(thisform.text2.text))=0) OR;
(len(alltrim(thisform.text3.text))=0))
thisform.commandgroup1.command1.enabled = .F.
endif

Crearea și exploatarea bazelor de date relaționale

3. Se construiește formularul form2:

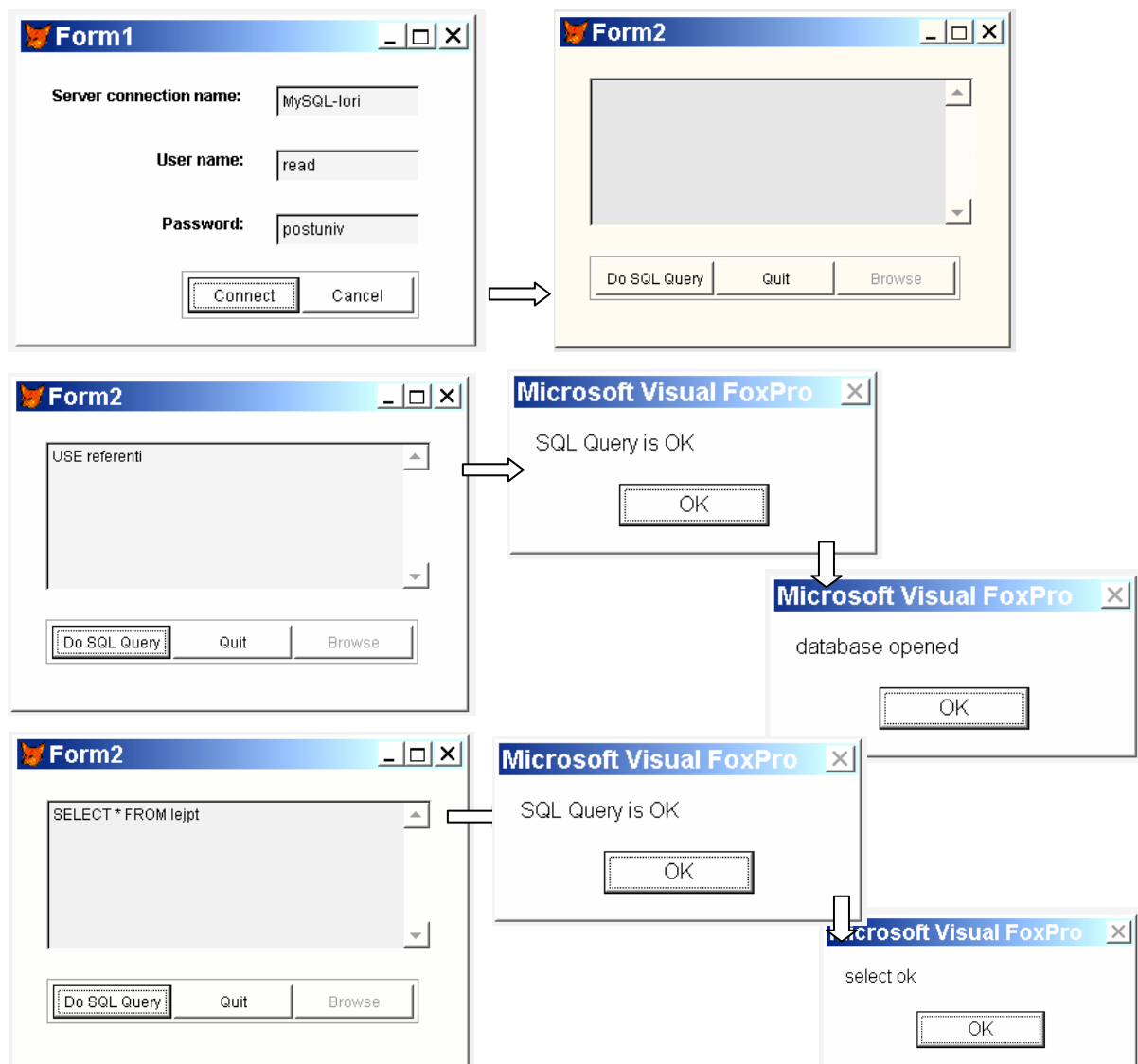
```
a = alltrim(thisform.edit1.text)
b = SQLEXP(nCH, a, 'MyCursor')
if (b>0)
    messagebox("SQL Query is OK")
    if upper(substr(a,1,3))='USE'
        messagebox("database opened")
        thisform.commandgroup1.command3.enabled = .F.
    endif
    if upper(substr(a,1,6))='SELECT'
        messagebox("select ok")
        thisform.commandgroup1.command3.enabled = .T.
    endif
endif
```

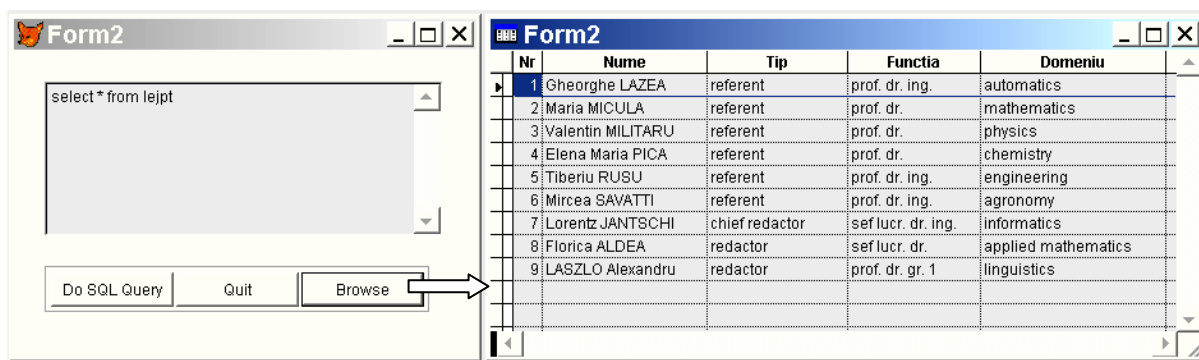


4. Se realizează programul principal (program1):

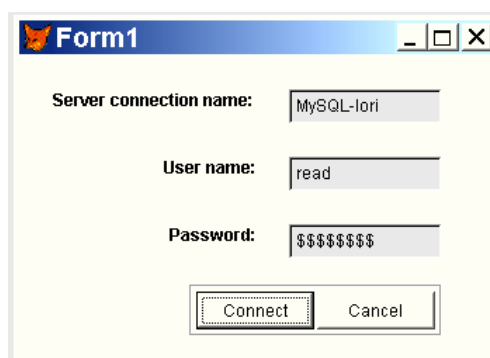
```
public nCH
do form form1.scx
read events
```

5. Execuția programului se face în modul următor (Program Manager – proj1/code/programs/program1/Run):





6. Se poate modifica formularul form1 astfel încât să nu afișeze parola la intrare. Se modifică proprietatea *PasswordChar* a casetei Text1: \$;
7. Se execută din nou aplicația:



33. Comenzi SQL pentru accesul la un server de date

În aplicațiile client/server sunt utile următoarele comenzi VFP:

CURSORSETPROP(cProperty [, eExpression] [, cTableAlias | nWorkArea])

Exemplu 1.

Următorul exemplu demonstrează cum se poate deschide *optimistic* o tabelă cu ajutorul funcției *CursorSetProp()*. Funcția se poate folosi inclusiv pentru tabele locale. Se folosește tabela *institutii* din baza de date *universitati*. Variabila de mediu *MultiLocks* se va seta pe ON, ca o necesitate pentru accesul prin intermediul tampoanelor pe disk. Tabela *institutii* din baza de date *universitati* este deschisă, iar funcția *CursorSetProp()* este folosită pentru a seta modul de acces optimist prin tampoane pe disk (*optimistic buffering mode*). Este afișat un mesaj informativ despre rezultatul operației.

```

CLOSE DATABASES
CLEAR
SET MULTLOCKS ON
OPEN DATABASE ('universitati')
USE institutii
!Success=CURSORSETPROP("Buffering", 5, "institutii")
IF !Success = .T.
    =MESSAGEBOX("Operation successful!",0,"Operation Status")
ELSE
    =MESSAGEBOX("Operation NOT successful!",0,"Operation Status")
ENDIF
    
```

Crearea și exploatarea bazelor de date relaționale

Lista parametrilor poate accepta mai multe proprietăți urmate de valorile lor, așa cum rezultă din următorul tabel:

proprietate (Property)	Valoarea (<i>eExpression</i>)
BatchUpdateCount (*)	Numărul de comenzi update trimise către sursa de date pentru tabelele tampon.
Buffering	1 – setează tamponul pentru înregistrări și tabelă la Off; 2 – setează modul <i>pesimistic</i> pentru tamponul la înregistrări la On; 3 – setează modul <i>optimistic</i> pentru tamponul la înregistrări la On; 4 – setează modul <i>pesimistic</i> pentru tamponul la tabelă la On; 5 – setează modul <i>optimistic</i> pentru tamponul la tabelă la On;
CompareMemo	.T. face ca câmpurile <i>Memo</i> , <i>General</i> și <i>Picture</i> să fie incluse în clauza <i>Where</i> pentru modificări; .F. – nu;
FetchAsNeeded	.F. – pentru vederi la distanță numărul înregistrărilor preluate e determinat de proprietatea <i>MaxRecord</i> ; pentru vederile locale toate înregistrările sunt preluate; .T. – toate;
FetchMemo (*)	.T. – preia și câmpurile memo; .F. – nu;
FetchSize (*)	Numărul de înregistrări citite progresiv de la distanță; implicit 100; -1 face ca să se preia întregul set;
KeyFieldList	Lista de câmpuri (specificate prin nume) care se dorește a fi preluate în cursor, separate prin virgulă;
MaxRecords (*)	Numărul maxim de înregistrări preluate; -1: toate; 0: nici una (util pentru adăugări, când se dorește execuția vederii, dar nu se dorește citirea de date de la distanță ci doar scrierea);
Prepared	.T. atunci când se face doar o operație de pregătire a interogării (vezi aplicația de la capitolul 32, în care s-a folosit funcția <i>SQLPrepare()</i> , similară ca efect); se folosește apoi <i>REQuery()</i> ;
SendUpdates	.T. sau .F.
Tables	Lista tabelelor la distanță separate prin virgulă
UpdatableFieldList	Lista câmpurilor pentru Update (nume câmp în tabela la distanță, urmat de nume câmp în cursor);
UpdateNameList	Numele câmpurilor în cursor (pentru redenumiri);
UpdateType	1 – vechile date sunt modificate cu cele noi; 2 – ștergerea datelor vechi și inserarea celor noi;
UseMemoSize (*)	Dimensiunea (în octeți) maximă a unui câmp de la care acesta

	când este preluat este stocat într-un câmp memo;
WhereType	Clauza Where pentru modificarea tabelelor la distanță; valori: 1 sau DB_KEY: doar câmpurile specificate în KeyFieldList; 2 sau DB_KEYANDUPDATABLE: câmpurile specificate în KeyFieldList și cele modificabile; 3 sau DB_KEYANDMODIFIED: câmpurile specificate în KeyFieldList și cele modificate; 4 sau DB_KEYANDTIMESTAMP: câmpurile specificate în KeyFieldList și cele accesate;

(*) aceste proprietăți au ca destinație primară utilizarea lor la vederi la distanță; setarea acestora pentru vederile locale nu are efect.

Dicționar

pessimistic buffering: previne ca un utilizator în sistem multi-user să acceseze o anume înregistrare sau o tabelă când un alt utilizator face schimbări asupra ei; asigură cea mai mare securitate de mediu pentru modificarea înregistrărilor dar încetinește operațiile cu utilizatorii;

optimistic buffering: o cale eficientă de a modifica înregistrări, deoarece blocarea acestora (lock) intervine doar atunci când acestea sunt scrise, și face ca să se minimizeze timpul în care un utilizator monopolizează sistemul într-un sistem multi-user; când se folosește blocarea la nivel de înregistrare sau tabelă în sistem multi-user, VFP aplică blocarea *optimistic*;

SQLCONNECT([DataSourceName, cUserID, cPassword | cConnectionName])

Exemplu 2.

```
STORE SQLCONNECT('MyFoxSQLNT', 'sa') TO gnConnHandle
IF gnConnHandle <= 0
    = MESSAGEBOX('Cannot make connection', 16, 'SQL Connect Error')
ELSE
    = MESSAGEBOX('Connection made', 48, 'SQL Connect Message')
    = SQLDISCONNECT(gnConnHandle)
ENDIF
```

SQLTABLES(nConnectionHandle [, cTableTypes] [, cCursorName])

Exemplul 3.

```
STORE SQLCONNECT('MyFoxSQLNT', 'sa') TO gnConnHandle
IF gnConnHandle < 0
    = MESSAGEBOX('Cannot make connection', 16, 'SQL Connect Error')
ELSE
    = MESSAGEBOX('Connection made', 48, 'SQL Connect Message')
    STORE SQLTABLES(gnConnHandle, 'TABLE', 'mycursor') TO nTables
    IF nTables = 1
        SELECT mycursor
        LIST
    ENDIF
ENDIF
```

SQLGETPROP(nConnectionHandle, cSetting)

Exemplul 4.

```
* Clear environment
CLOSE ALL
CLEAR ALL
CLEAR
* Display the Select Connection or Data Source dialog box
* to choose a connection
nHandle=SQLCONNECT()
* Test connection, report results
IF nHandle > 0
  cSource= SQLGETPROP(nHandle, "datasource")
  =MESSAGEBOX("Current Data Source = "+cSource,0,"Connection Results")
ELSE
  =MESSAGEBOX("Connection Error = " + ;
    ALLTRIM(STR(nHandle)),0,"Connection Results")
ENDIF
=SQLDISCONNECT(nHandle)
```

SQLSETPROP(nConnectionHandle, cSetting [, eExpression])

Exemplul 5.

```
* Clear environment
CLOSE ALL
CLEAR ALL
CLEAR
* Display the Select Connection or Datasource dialog box
* to choose a connection
nHandle=SQLCONNECT()
* Test connection, report results
IF nHandle > 0
  * Set PacketSize
  nSet=SQLSETPROP(nHandle, "PacketSize", 2048 )
  * Test setting and display results
  IF nSet > 0
    =MESSAGEBOX("PacketSize was set to 2048",0,"Connection Results")
  ELSE
    =MESSAGEBOX("Error setting PacketSize",0,"Connection Results")
  ENDIF
ELSE
  =MESSAGEBOX("No Connection",0,"Connection Results")
ENDIF
=SQLDISCONNECT(nHandle)
```

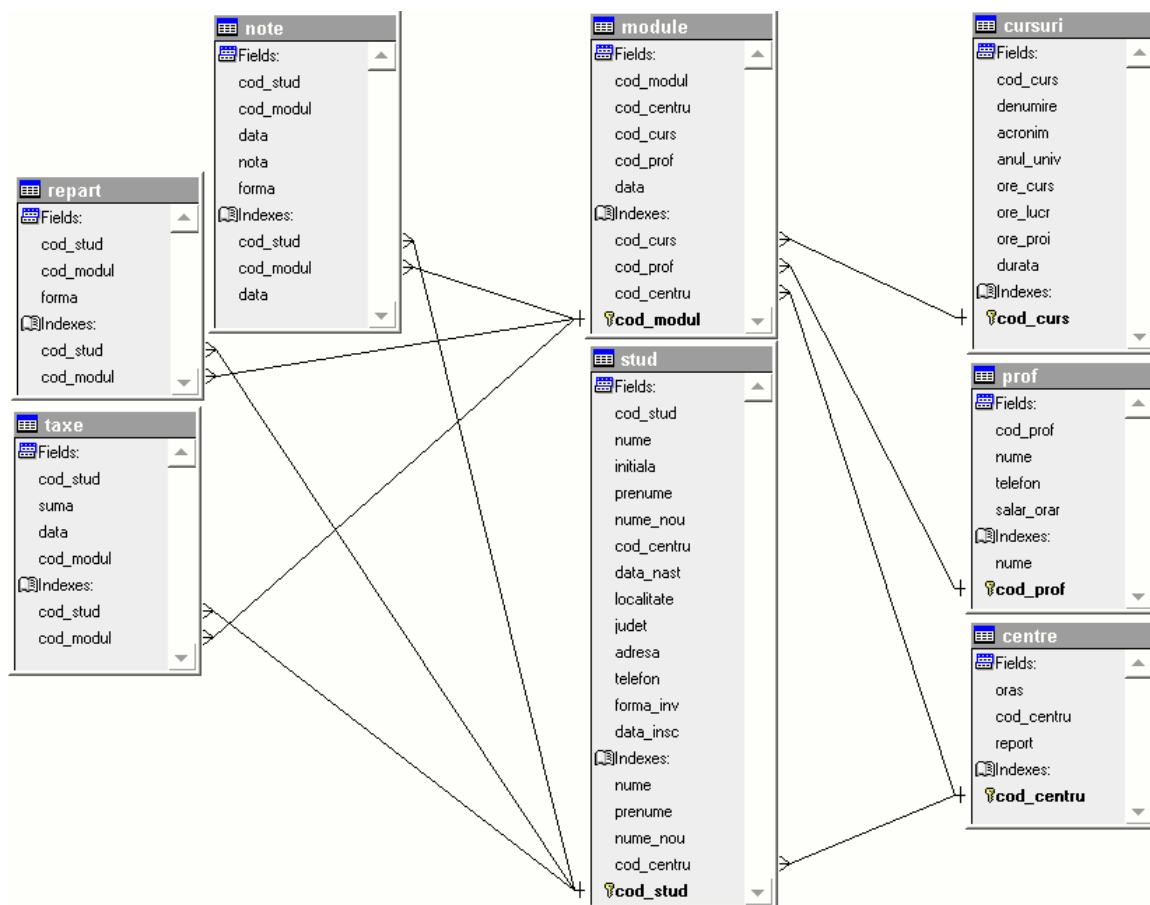
SQLMORERESULTS(nConnectionHandle)

Exemplul 6.

```
=SQLSETPROP(gnConnHandle, 'BatchMode', .F.) && Individual result sets
=SQLEXP(gnConnHandle, 'SELECT * FROM authors;
  SELECT * FROM titles')
=SQLMORERES(gnConnHandle) && First result set
=SQLMORERES(gnConnHandle) && Second result set
```

33. Aplicație exemplu

Se consideră baza de date cu cursanții Școlii Academice Postuniversitare de Informatică Aplicată și Programare cu următoarea structură:



Obiectivul este crearea interfeței de exploatare a bazei de date. după crearea bazei de date, a tabelor și stabilirea relațiilor între tabele, conținutul directorului cu baza de date este:

↑Name	Ext	Size
BD_PU	dbc	23,983
Bd_pu	dct	4,608
Bd_pu	dcx	10,240
centre	CDX	3,072
centre	dbf	752
cursuri	CDX	3,072
cursuri	dbf	2,354
Module	cdx	9,216
module	dbf	1,790
Note	cdx	26,624
note	dbf	40,342
Prof	cdx	6,144
prof	dbf	1,574
repart	CDX	16,896
repart	dbf	20,048
Stud	cdx	41,472
stud	dbf	140,554
stud	FPT	28,234
Taxe	cdx	35,328
taxe	dbf	118,628

Se creează acum formularele și rapoartele:

Crearea și exploatarea bazelor de date relaționale

1. Formulare pentru adăugarea și modificarea unui centru, și curs; tabelele nu au chei străine deci formularele sunt pe baza unei singure tabeli; se merge din meniu pe calea: New/Form/Wizard/Form Wizard; se generează astfel cele 3 formulare:

The first two screenshots show the 'Form Wizard' dialog box for creating a form for the 'CENTRE' table. Step 1 'Select Fields' shows the 'Available fields' list with 'Oras', 'Cod_centru', and 'Report' selected. Step 3 'Sort Records' shows the 'Selected fields' list with 'Cod_centru' selected for sorting in ascending order.

The next two screenshots show the 'Form Wizard' dialog box for creating a form for the 'CURSURI' table. Step 1 'Select Fields' shows the 'Available fields' list with 'Cod_curs', 'Denumire', 'Acronim', 'Anul_univ', 'Ore_curs', 'Ore_lucr', 'Ore_proi', and 'Durata' selected. Step 3 'Sort Records' shows the 'Selected fields' list with 'Anul_univ' and 'Cod_curs' selected for sorting in ascending order.

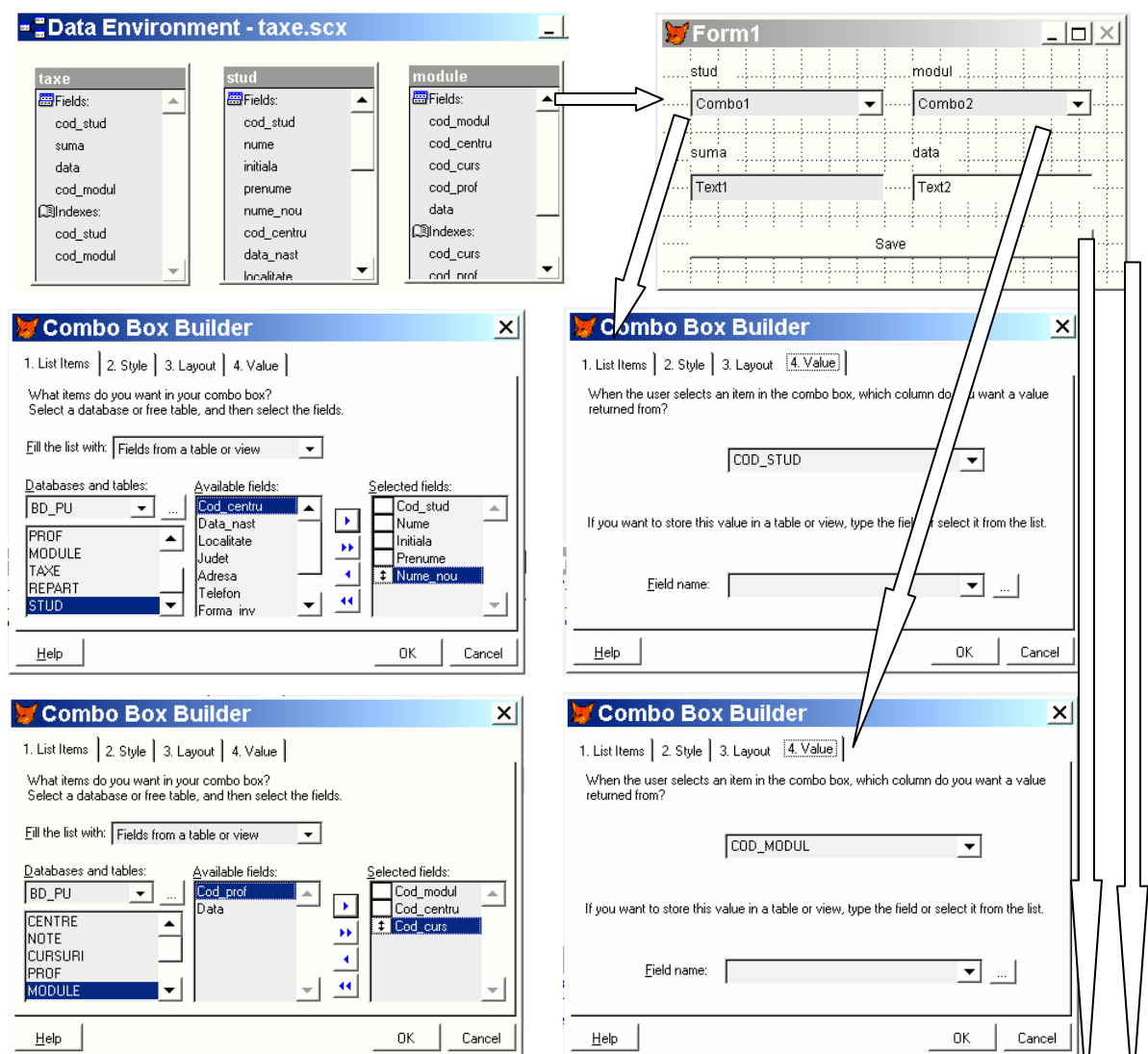
2. Se salvează aceste formulare în directorul cu baza de date. Se creează formulare pentru adăugarea/modificarea unui student; tabela are cheie străină din tabela centre; se folosește wizard-ul pentru aceasta: New/Form/Wizard/One-to-Many Form Wizard;

The first two screenshots show the 'One-to-Many Form Wizard' dialog box for creating a form for the 'STUD' table. Step 1 'Select Parent Table Fields' shows the 'Available fields' list with 'Report' selected. Step 2 'Select Child Table Fields' shows the 'Available fields' list with 'Cod_stud', 'Nume', 'Initiala', 'Prenume', 'Nume_nou', 'Cod_centru', 'Data_nast', and 'Localitate' selected.

The next two screenshots show the 'One-to-Many Form Wizard' dialog box for creating a form for the 'STUD' table. Step 3 'Relate Tables' shows the 'CENTRE (BD_PU)' table with 'cod_centru' selected and the 'STUD (BD_PU)' table with 'cod_centru' selected. Step 5 'Sort Records' shows the 'Selected fields' list with 'Oras' selected for sorting in ascending order.

Crearea și exploatarea bazelor de date relaționale

3. Tabelele note, repart și taxe sunt tabele ce implementează relații (m,n) între tabelele module și stud; adăugarea unei note presupune selecția studentului din tabela de studenți, selectarea modulului din tabela de module și introducerea datei, formei și notei; adăugarea unei repartiții presupune selecția studentului din tabela de studenți, selecția modulului din tabela de module și introducerea formei; adăugarea unei taxe presupune selecția studentului din tabela stud, selecția modulului din tabela de module și introducerea sumei și datei; se merge pe calea New/Form/New File;

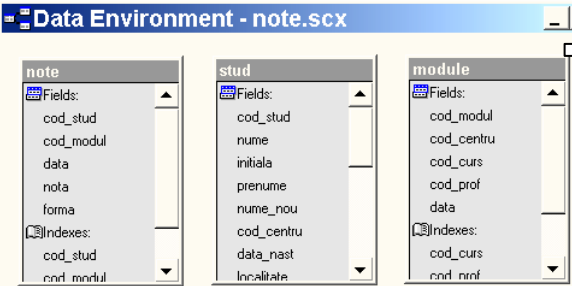
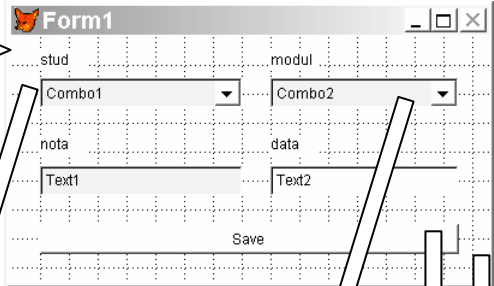
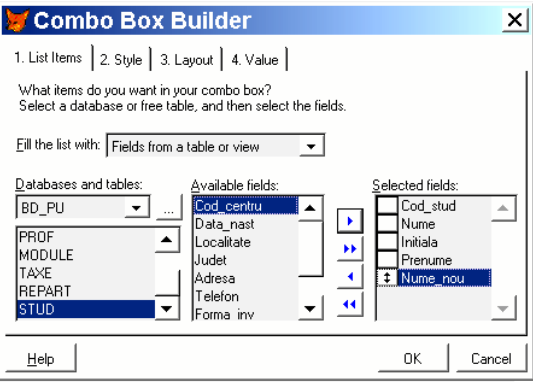
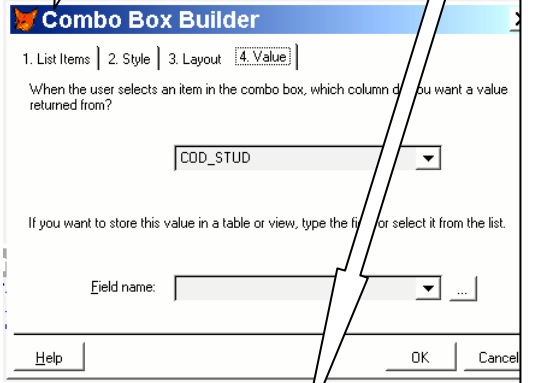
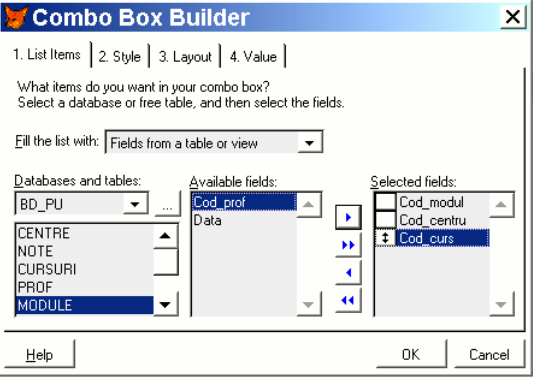
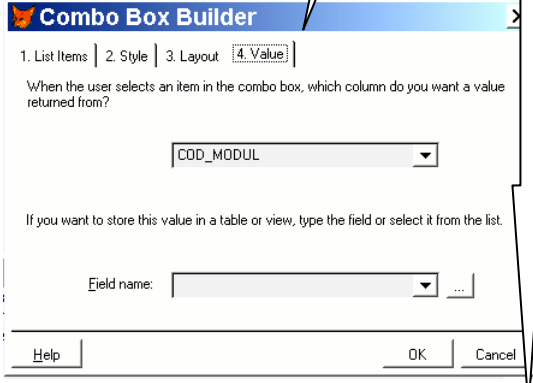


```
select "stud.dbf"
c_stud = cod_stud
select "module.dbf"
c_modul = cod_modul
select "taxe.dbf"
append blank
repl taxe.cod_modul with c_modul,;
taxe.cod_stud with c_stud,;
taxe.data with date(),
taxe.suma with val(thisform.text1.text)
```

```
Form1.activate:
thisform.text2.text = dtoc(date())
```

4. La fel se procedează și pentru celelalte două formulare:

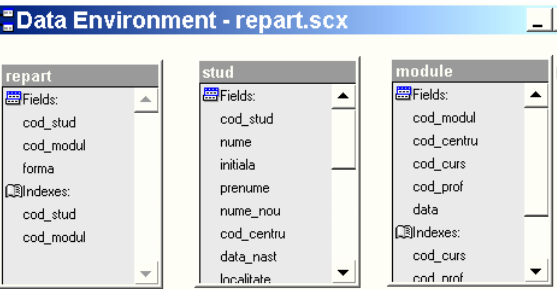
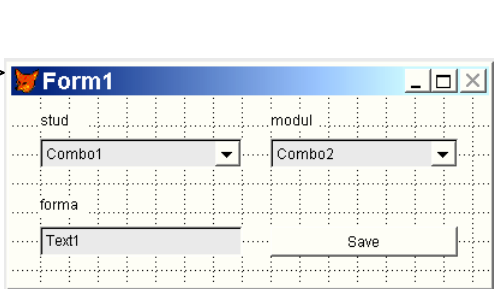
Crearea și exploatarea bazelor de date relaționale

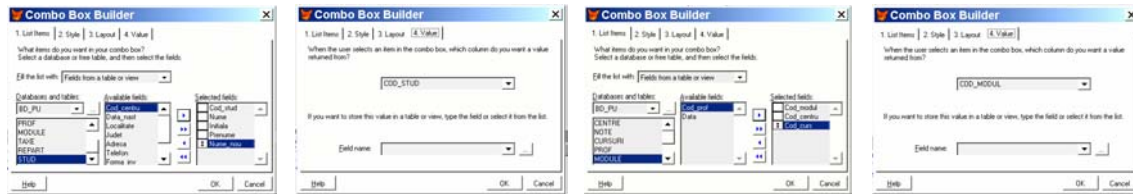
```

select "stud.dbf"
c_stud = cod_stud
select "module.dbf"
c_modul = cod_modul
select "note.dbf"
append blank
repl note.cod_modul with c_modul,;
note.cod_stud with c_stud,;
note.data with date(),
note.nota with val(thisform.text1.text)
    
```

Form1.activate:
thisform.text2.text = dtoc(date())

Crearea și exploatarea bazelor de date relaționale



Command1.Caption: Save;

Command1.Click:

```
select "stud.dbf"
```

```
c_stud = cod_stud
```

```
select "module.dbf"
```

```
c_modul = cod_modul
```

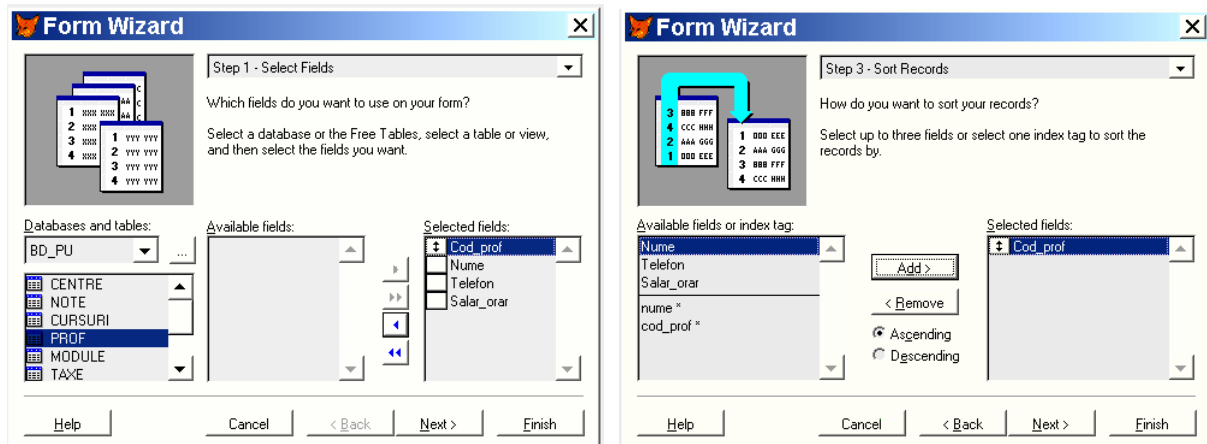
```
select "repart.dbf"
```

```
append blank
```

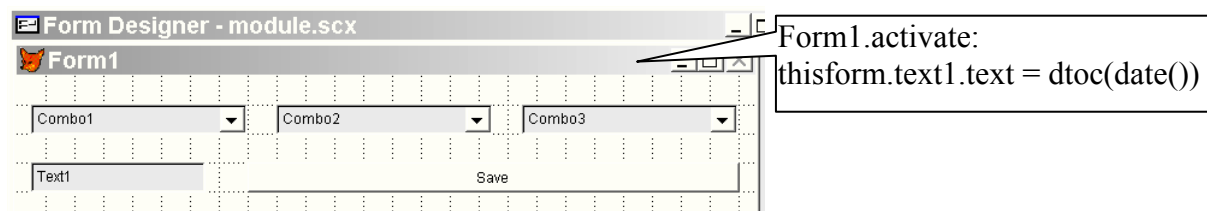
```
repl repart.cod_modul with c_modul, repart.cod_stud with c_stud,;
```

```
repart.forma with alltrim(thisform.text1.text)
```

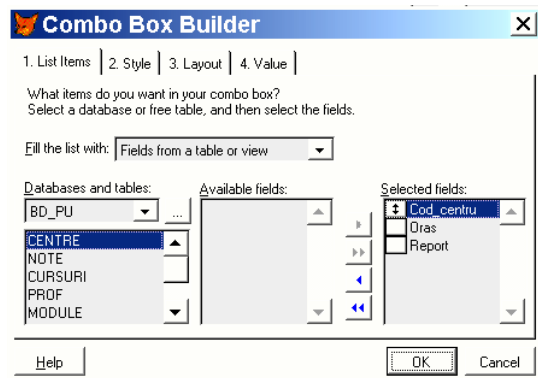
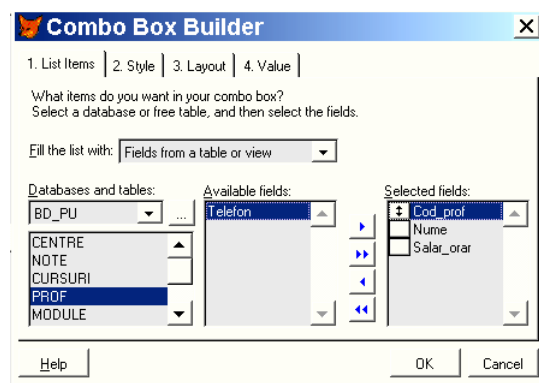
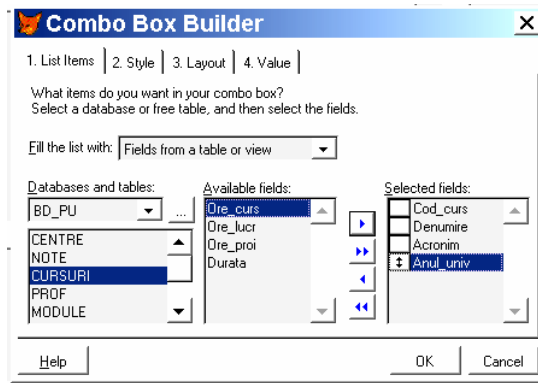
5. Formularul pentru introducerea datelor în tabelele prof și module poate fi făcut cu ajutorul wizard-ului: New/Form/Wizard/One-to-Many Form Wizard, la fel ca la formularul stud, însă s-ar pierde o legătură; se poate face în forma anterioară, se crează un formular pentru prof:



după care se creează un formular pentru modul:

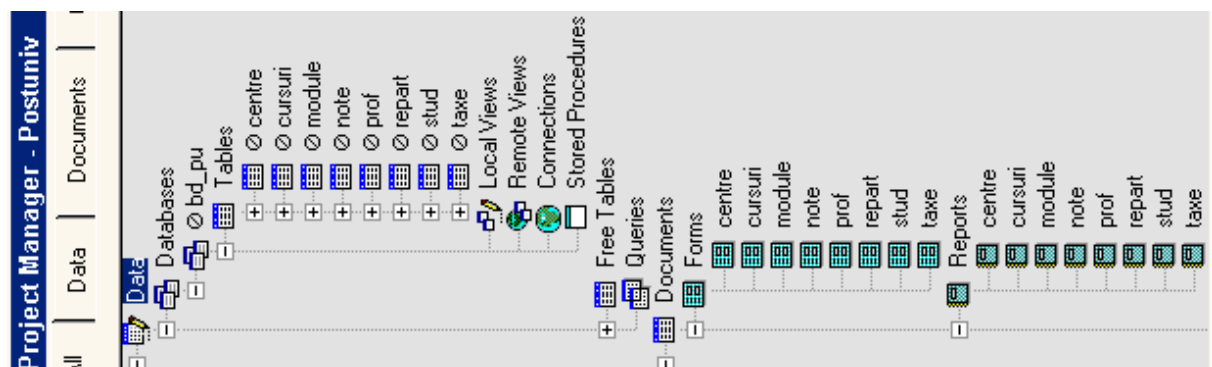
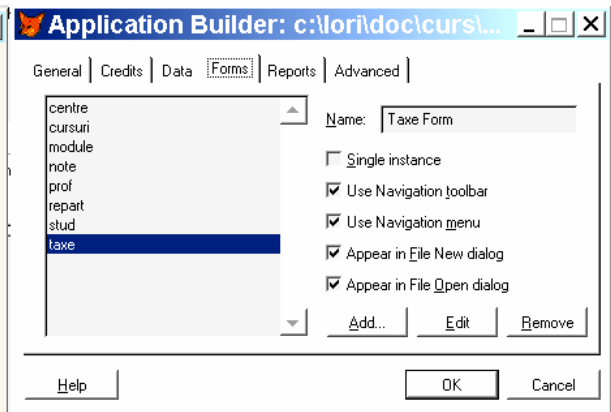
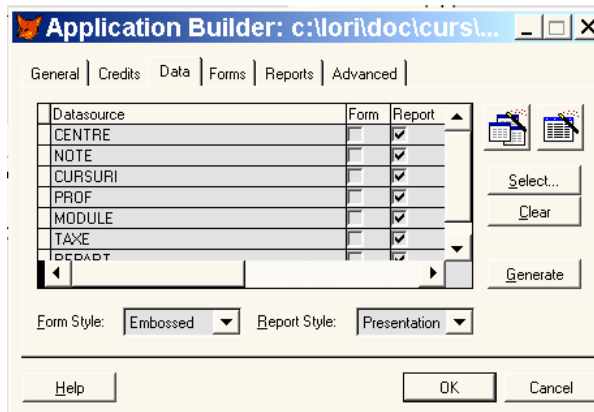


Crearea și exploatarea bazelor de date relaționale



```
select "cursuri.dbf"
cod_c = cursuri.cod_curs
select "prof.dbf"
cod_p = prof.cod_prof
select "centre.dbf"
cod_n = centre.cod_centru
select module
append blank &&are functie autoincrement
replace module.cod_centru with cod_n,;
      module.cod_curs with cod_c,;
      module.cod_prof with cod_p,;
      module.data with date()
```

6. Se creează rapoartele cu ajutorul lui Report Wizard;
7. Se integrează aplicațiile într-n proiect cu ajutorul lui Application Wizard:
New/Project/Wizard:



34. Documentarea aplicațiilor windows cu Microsoft HTML Help

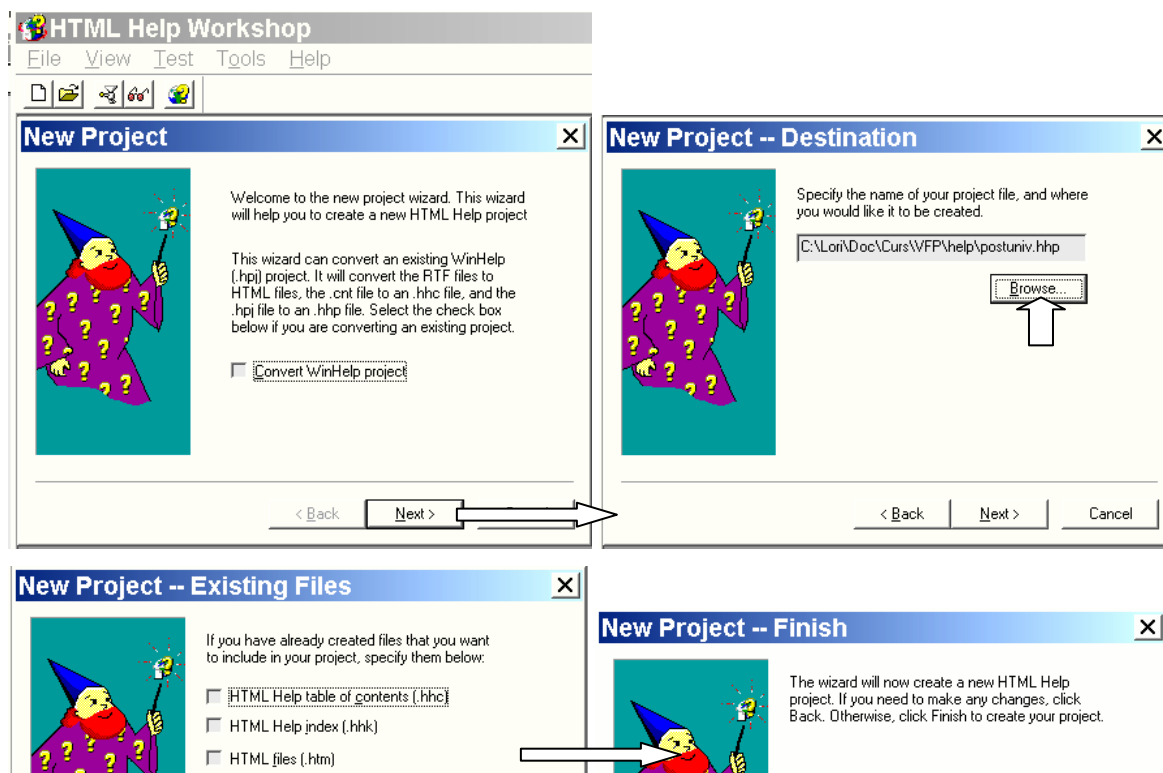
Microsoft® HTML Help constă dintr-un program de vizualizare online *Help Viewer* care folosește componentele lui *Microsoft Internet Explorer* pentru a afișa conținutul help-ului. Suportă HTML, ActiveX®, Java™, limbaje de scriptare (JScript®, and Microsoft Visual Basic® Scripting Edition) și imagini în format HTML (.jpeg, .gif, și .png).

HTML Help Workshop este instrumentul cu care se pot crea și exploata proiectele help și fișierele auxiliare.

HTML Help project este un fișier text cu extensia .hhp care organizează toate elementele unui sistem help. El conține legăturile către toate topicurile HTML (fișiere cu extensiile .html și .htm), imagini (.jpeg, .gif, .png), index (.hhk), și cuprins (.hhc). *HTML Help Workshop* combină apoi toate aceste fișiere pentru a genera un singur fișier cu extensia .chm.

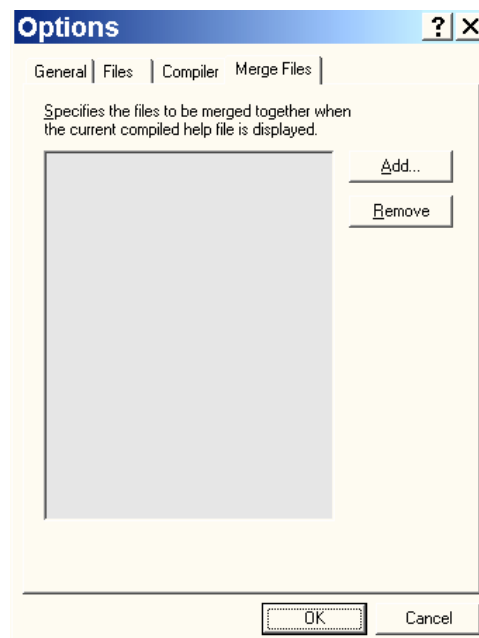
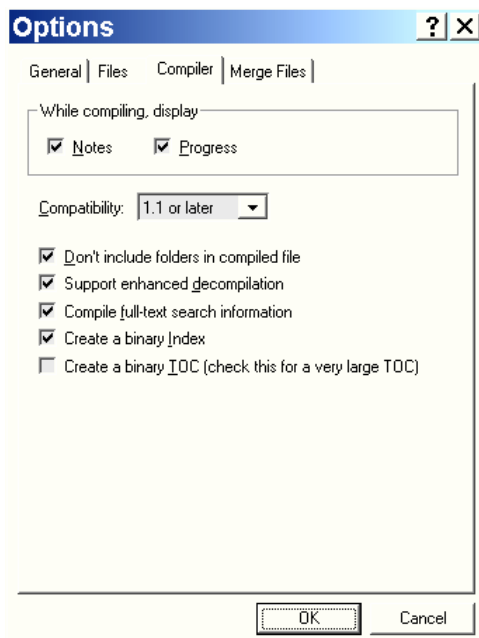
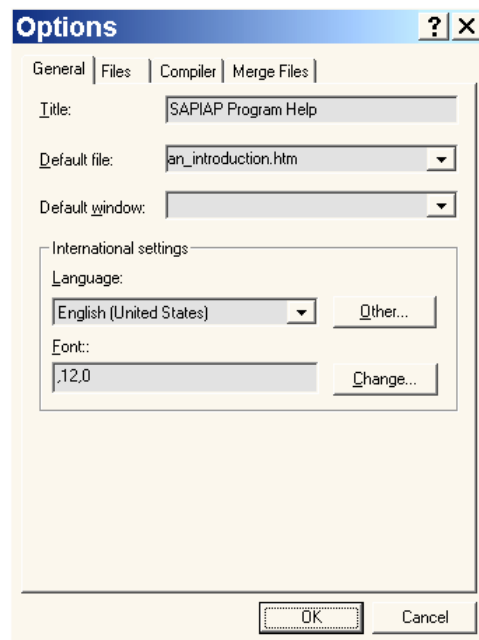
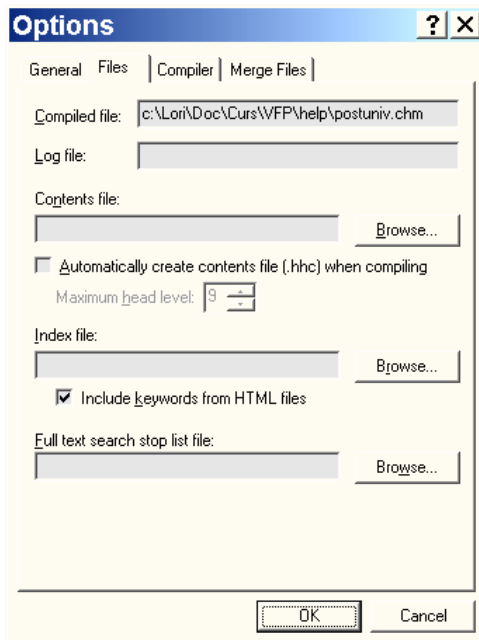
35. Crearea unui nou fișier help (.chm) cu HTML Help Workshop

HTML Help Workshop este un produs free Microsoft și poate fi descărcat de la adresa: <http://lori.east.utcluj.ro/JLorentz/free/htmlhelp.exe>. Se instalează în sistem (sunt necesare privilegii de administrator) după care se încarcă în execuție pe calea *Start/Programs/HTML Help Workshop/HTML Help Workshop*. Se creează un nou help (File/New). Dacă au fost deja create fișiere componente ale proiectului, se pot include în el:



După generarea proiectului, se pot defini opțiunile acestuia, în care se includ topicurile, așa cum rezultă din figurile următoare.

Crearea și exploatarea bazelor de date relaționale



Până în acest moment, conținutul fișierului *postuniv.hhp* este:

[OPTIONS]

Auto Index=Yes

Compatibility=1.1 or later

Compiled file=postuniv.chm

Default Font=, 12,0

Default topic=data\an_introduction.htm

Display compile progress=Yes

Enhanced decompilation=Yes

Flat=Yes

Full-text search=Yes

Language=0x409 English (United States)

Title=SAPIAP Program Help

[FILES]

data\t_stud.htm

data\an_introduction.htm

Crearea și exploatarea bazelor de date relaționale

f_centre.htm
f_cursuri.htm
f_module.htm
f_note.htm
f_prof.htm
f_stud.htm
r_centre.htm
r_cursuri.htm
r_module.htm
r_note.htm
r_prof.htm
r_stud.htm
t_centre.htm
t_cursuri.htm
t_module.htm
t_note.htm
t_prof.htm
a_documentation.htm
[INFOTYPES]

Fișierele *.htm sunt fișiere (D)HTML standard și pot fi generate cu orice program creator de pagini web (Netscape Composer, Microsoft Word, FrontPage).

Iată conținutul fișierelor an_introduction.htm, a_documentation.htm și f_centre.htm:

an_introduction.htm

```
<html><head><title>SAPIAP Help</title></head><body>  
Baza de date a Scolii Academice Postuniversitare de Informatica Aplicata si Programare este  
in urmatoarea structura:<BR>  
<BR>  
<BR>Vezi si:<BR>  
tabela <A HRef = "t_centre.htm">centre</A><BR>  
tabela <A HRef = "t_stud.htm">studenti</A><BR>  
tabela <A HRef = "t_cursuri.htm">cursuri</A><BR>  
tabela <A HRef = "t_prof.htm">profesori</A><BR>  
tabela <A HRef = "t_module.htm">module</A><BR>  
tabela <A HRef = "t_note.htm">note</A><BR>  
informatiile despre realizarea <A HRef = "a_documentation.htm">helpului</A>  
</body></html>
```

a_documentation.htm

```
<html><head><title>SAPIAP Help</title></head><body>  
Fișierele incluse in topicul acestui help sunt:<BR>  
<BR>  
</body></html>
```

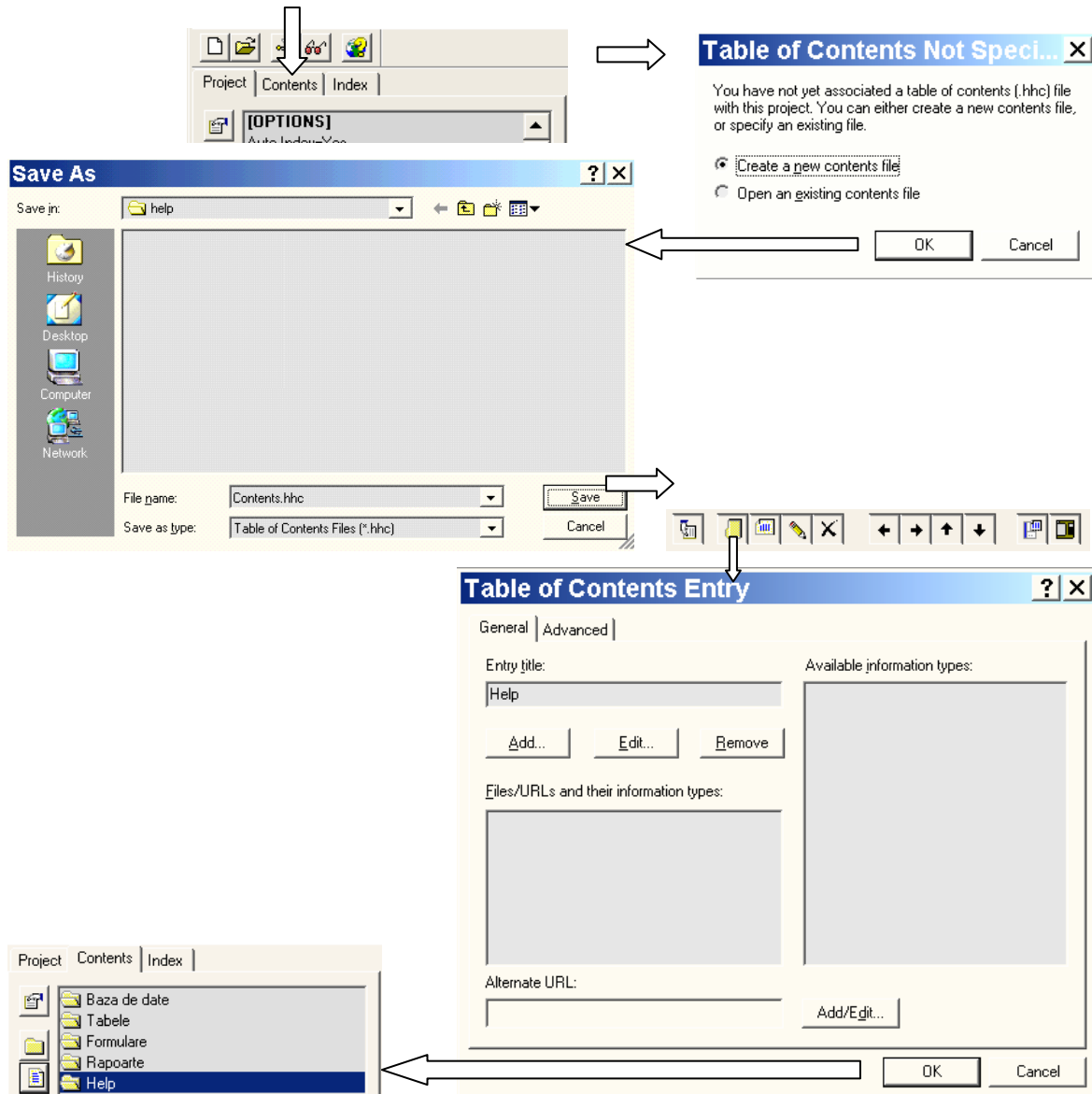
f_centre.htm

```
<html><head><title>Formularul centre</title></head><body>  
<BR>Formularul Centre ne permite sa prelucreaza informatiile din tabela de centre de  
invatamant:<BR>  
<BR>  
<BR>Parcurgerea se realizeaza cu butoanele:<BR>
```

Crearea și exploatarea bazelor de date relaționale

```
<ul><li>top: inceputul tabelii</li>prev: inregistrarea anterioara</li>next: urmatoarea inregistrare</li>bottom: ultima inregistrare</ul><BR><BR>Cautarea informatiilor se face cu butonul Find:<BR><BR><BR>Tiparirea informatiilor se face cu butonul print:<BR><BR><BR>Adaugarea unui nou centru se face cu butonul Add:<BR><BR><BR>Editarea informatiilor despre un centru se face cu butonul Edit:<BR><BR><BR>Stergerea unui centru se face cu butonul Delete iar iesirea din formular se face cu butonul Exit.<BR>Vezi si:<BR>tabela <A HRef = "t_centre.htm">centre</A><BR>raportul <A HRef = "r_centre.htm">centre</A><BR>pagina de<A HRef = "an_introduction.htm">inceput</A><BR></body></html>
```

Se poate acum crea cuprinsul, începând cu repertoarul acestuia:



Crearea și exploatarea bazelor de date relaționale

Se poate schimba iconița implicită asociată unei intrări pe calea:

Contents/Edit Selection/Table of Contents Entry/Advanced. Legenda este alăturată:

Table of Contents Entry

General | Advanced

Window:

Frame:

Comment:

☐ Mark as a new entry

☐ Change entry to a page

Image index: 

OK Cancel

1		15		29	
2		16		30	
3		17		31	
4		18		32	
5		19		33	
6		20		34	
7		21		35	
8		22		36	
9		23		37	
10		24		38	
11		25		39	
12		26		40	
13		27		41	
14		28		42	

Se pot crea legături către pagini în cuprins pe calea:

Table of Contents Entry

General | Advanced

Entry title:

Add... Edit... Remove

Files/URLs and their information types:

Alternate URL:

Add/Edit... OK Cancel

Baza de date

Structura

Tabele

Centre

Module

Note

Profesori

Studenti

Microsoft Visual FoxPro



Datorita conventiilor de nume HTML Help Workshop nu foloseste pentru identificarea topicului decât primele 4 litere ale fișierelor din topic. Nu folosiți subdirectoare și nu folosiți nume cu primele 4 litere identice pentru fișierele din proiectul dvs *.hlp

OK

În mod implicit se generează o fereastră de help cu puține controale înglobate. Se poate însă crea o fereastră definită de utilizator. Se merge pe calea:

Window Types

Navigation Pane | Buttons | Styles | Extended Styles

General

Window type:

Add... Remove

Title bar text:

OK Cancel

Window Types

Navigation Pane | Buttons | Styles | Extended Styles

General

Window type:

Button Types

☒ Hide/Show ☒ Refresh ☒ Locate

☒ Back ☒ Home ☐ Jump 1

☒ Forward ☒ Options ☐ Jump 2

☒ Stop ☒ Print

Jump 1 text:

Jump 2 text:

OK Cancel

Window Types

Navigation Pane | Buttons | Styles | Extended Styles

General

Window type:

Window size and position

☒ Save user defined window position after first use

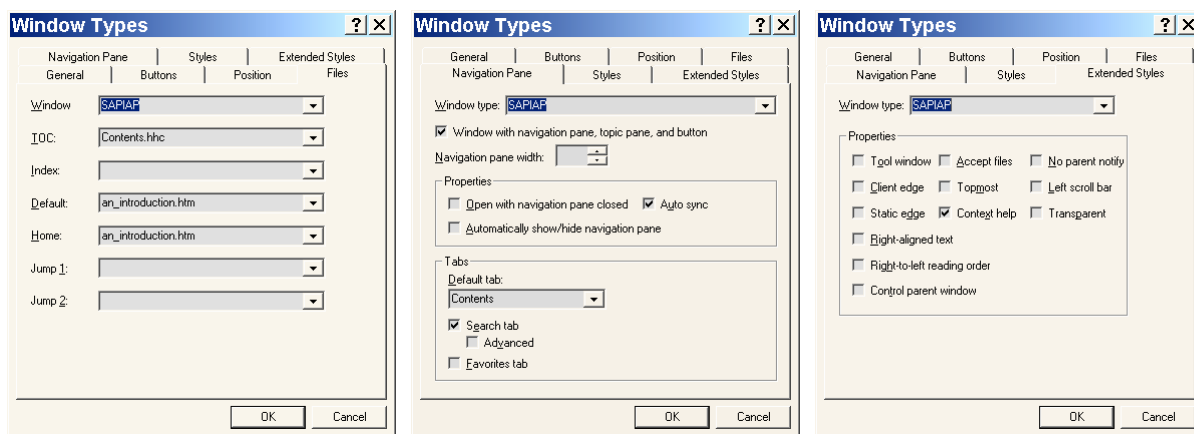
Left: Width:

Top: Height:

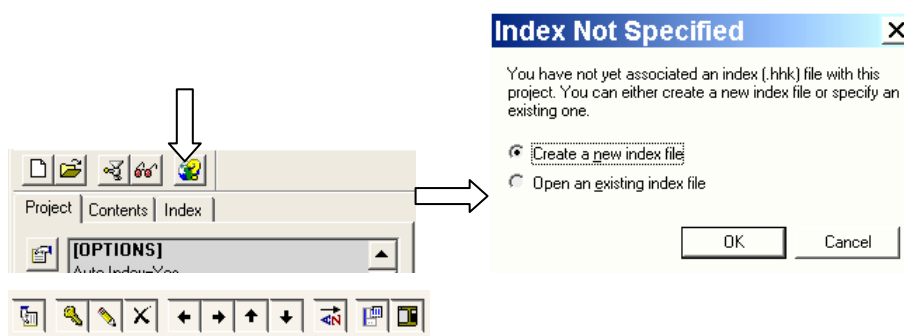
Autogizer Default Positions

OK Cancel

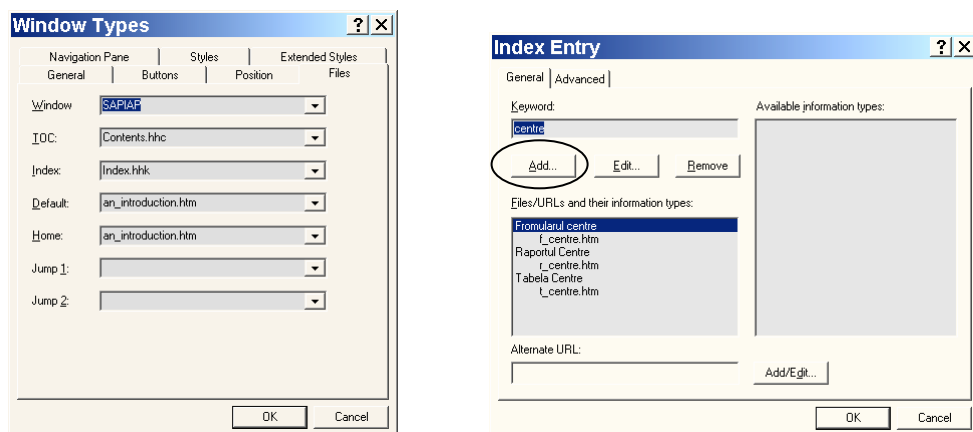
Crearea și exploatarea bazelor de date relaționale



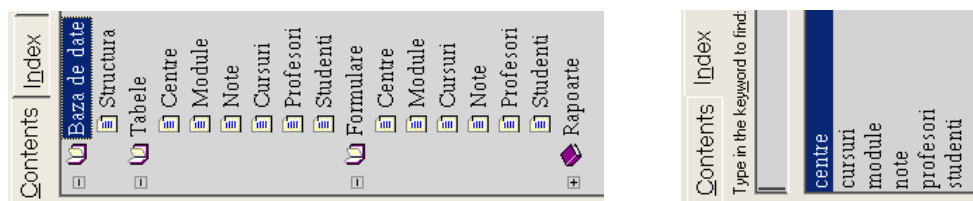
Se poate acum crea indexul:



Se adaugă opțiunea de index în Window Types (*Add/Modify Window definitions*). Se adaugă apoi noi intrări în index pe aceeași cale ca pentru paginile din cuprins:



Compilarea helpului are ca rezultat fișierul `postuniv.chm`. Încărcarea acestuia în execuție duce la:



Se poate acum regenera aplicația cu *Application Builder* când se specifică fișierul `help postuniv.chm` în pagina *Advanced* a acestuia.

36. Test de *Crearea și Exploatarea Bazelor de Date Relaționale*

Se consideră o bază de date care conține două tabele relatate pe baza unei relații de tipul 1 la n (One to Many Relationship). Tabela cu cheia primară a relației se va numi tabela părinte iar tabela cu cheia străină a relației se va numi tabela fiu.

1. Să se realizeze o vedere cu parametru care să conțină un câmp de identificare din tabela părinte (de preferință o cheie candidată) și cel puțin două câmpuri din tabela fiu în la care căutarea să se facă într-un câmp din tabela fiu după o valoare de tip caracter. Ordonarea să se facă după valorile câmpului din tabela părinte și apoi după valorile unui câmp din tabela fiu.
2. Să se realizeze o interogare pe baza tabelii părinte și tabelii fiu care să conțină două câmpuri din tabela părinte și un câmp care să conțină numărul înregistrărilor relatate pe baza relației din tabela fiu pentru fiecare înregistrare din tabela părinte. Să se ordoneze înregistrările după un câmp din tabela părinte. Să se denumească câmpul ce conține numărul înregistrărilor din tabela fiu nrc.
3. Să se realizeze un raport care să conțină un câmp din tabela părinte și două câmpuri din tabela fiu mai puțin cheile străine și primare și să se grupeze informațiile după cheia primară din tabela părinte.
4. Să se realizeze un formular care să permită introducerea unei înregistrări în tabela fiu pe baza unei selecții a înregistrării relatate în tabela părinte cu ajutorul unei liste combo.
5. Să se realizeze un meniu pentru o aplicație.
6. Întrebări:
 - a. Ce este o cheie primară și o cheie străină;
 - b. Ce este o tabelă părinte și o tabelă fiu;
 - c. Ce este un index primar și ce este un index regular;
 - d. Caracterizați o tabelă, o vedere, o interogare și un cursor;
 - e. Cum implementați o relație m la n într-o bază de date;
 - f. Care sunt etapele unei conectări la un server de baze de date;
 - g. Care este diferența între un meniu sistem și un meniu contextual;
 - h. Cum se poate realiza un help pentru o aplicație;

37. Model de soluție pentru testul de Crearea și Exploatarea Bazelor de Date Relaționale

Fie baza de date *universități*. Soluție:

The image displays a series of screenshots from Microsoft Access, illustrating the design and implementation of a database solution for the 'universități' database.

Database Design:

- institutii** table: Fields include nume, acronim, adresa, web, nr, and aa.
- contacte** table: Fields include nume, functia, email, poz, nr, and aa.

Query Design:

- Query1**: A query showing the relationship between **institutii** and **contacte** tables. The criteria are: **institutii.nume** = **contacte.nume**.
- Query2**: A query showing the relationship between **institutii** and **contacte** tables. The criteria are: **institutii.nume** = **contacte.nume**.
- Query3**: A query showing the relationship between **institutii** and **contacte** tables. The criteria are: **institutii.nume** = **contacte.nume**.

Form Design:

- Form1**: A form with a Combo box and a Command button. The Combo box is labeled 'Nume' and the Command button is labeled 'Adauga'.
- Form2**: A form with a Combo box and a Command button. The Combo box is labeled 'Nume' and the Command button is labeled 'Adauga'.

Code Snippets:

```

Combo1.Click
Object: Combo1
Procedure: Click
thisform.command1.enabled = .t.
select contacte
set filter to alltrim(thisform.combobox1.text)
thisform.refresh

Command1.Click
Object: Command1
Procedure: Click
select contacte
append blank
replace nr with institutii.nr
    
```

Print Preview:

The Print Preview window shows the output of the query, displaying the names of the universities and their contact information.