Curs 12. Programarea formularelor şi rapoartelor
1. Programarea formularelor
2. Programarea rapoartelor

1. Programarea formularelor

Majoritatea aplicaţiilor Access pot fi privite ca fiind colecţii de formulare, deoarece o mare parte din funcţionalitatea unei aplicaţii se bazează pe acestea.
Fiecare formular şi raport Access are propriul modul în care se află procedurile pentru tratarea evenimentelor legate de lucrul cu formularul sau raportul respectiv sau cu controalele acestuia, precum şi alte proceduri pe care acestea le apelează. In unele cazuri, Access furnizează parametrii acestor proceduri, dându-le astfel informaţii despre circumstanţele procedurii evenimentului respectiv. (De exemplu, procedurile pentru tratarea evenimentelor legate de activitatea cu mouse-ul primesc informaţii despre poziţia cursorului şi despre apăsarea butoanelor). Acest gen de încapsulare (înglobarea într-un singur obiect atât a caracteristicilor ce ţin de aspectul formularului, cât şi a celor ce ţin de comportamentul lui), permite refolosirea unui formular în cadrul mai multor aplicaţii.
Pentru a deschide modulul unui formular, deschidem formularul în modul *Design View* şi alegem butonul **Code** de pe bara de instrumente.
O problemă pe care trebuie s-o avem în vedere este aceea că, prin copierea unui control de pe un formular pe altul, nu se vor copia şi procedurile pentru tratarea evenimentelor controlului respectiv în modulul formularului destinaţie.

Ferestre Windows şi formulare Access

Pentru a putea folosi cât mai eficient diferitele tipuri de formulare Access, trebuie să vedem funcţionalitatea conceptelor de identificator (*handle)* Windows, clasă fereastră, obiect de tip fereastră şi relaţiile dintre acestea şi dintre ferestre.

Identificatorul unei ferestre (Proprietatea *hWnd)*

In Windows, aproape orice obiect pe care îl vedem pe ecran (butoane, cutii de dialog, bare de derulare, bare de stare) este o fereastră. Pentru a putea gestiona ferestrele, Windows le atribuie câte un identificator unic. Acest identificator este una dintre proprietăţile unei ferestre şi deci şi a unui formular în Access. Nu putem avea totuşi acces la această proprietate prin intermediul paginii de proprietăţi, pentru că ea are altă valoare de fiecare dată când se deschide formularul. O putem, în schimb, obţine prin intermediul codului VBA, în timp ce formularul rulează în modul *Form View*.

Clase fereastră

Fiecare fereastră în Windows este un obiect al unei clase fereastră. Fiecărui tip de fereastră din Access îi corespunde deci o clasă fereastră. Intre diferitele ferestre din Access există legături de genul celor întâlnite într-o ierarhie: o fereastră-părinte poate avea mai multe ferestre-copil.

Interfaţa MDI

Majoritatea aplicaţiilor Windows prezintă o interfaţă standard, numită *Multiple Document Interface* (MDI). O astfel de interfaţă se caracterizează prin faptul că fereastra principală a aplicaţiei poate conţine mai multe ferestre subordonate, numite şi ferestre-copil. Microsoft Access este o astfel de aplicaţie (ca şi Microsoft Excel sau Microsoft Word, de exemplu).
Organizarea ierarhică a unei aplicaţii MDI este următoarea: există o fereastră principală, care conţine o fereastră specială numită "fereastra MDI client" şi care, la rândul ei, conţine mai multe ferestre copil.

Formulare normale, modale şi de tip popup

Termenii normal, modal şi popup descriu modul în care interacţionează un formular cu ferestrele altor formulare şi cu fereastra principală a aplicaţiei Access. Putem controla aceste atribute prin intermediul proprietăţilor *Popup* şi *Modal,* la crearea formularului sau prin intermediul acţiunii *OpenForm.*
Un formular normal permite altor formulare deschise în Access să se suprapună peste el şi să preia focusul. Formularele normale sunt ferestre copil ale ferestrei MDI client şi, de aceea, nu pot depăşi cadrul acesteia.
Un formular de tip popup este afişat deasupra tuturor ferestrelor deschise în Access. El nu este un copil al ferestrei MDI client, ci al ferestrei principale Access. De aceea, putem poziţiona aceste formulare şi în afara ferestrei Access şi le putem da şi o altă dimensiune mai mare decât a acesteia. Având în vedere fereastra principală Access ca părinte, dacă vom minimiza aplicaţia Access, se vor minimiza şi ele.
Un formular modal (fie că estesau nu de tip popup) păstrează focusul până când îl închidem sau dăm proprietăţii sale *Visible* valoarea No. Intre timp, Access ignoră orice alte acţiuni ale dumneavoastră care nu au legătură cu formularul.

Controlaţi comportamentul formularului cu ajutorul proprietăţilor

Access dă posibilitatea de a controla comportamentul unui formular prin intermediul ferestrei de proprietăţi.
Dacă formularul nu va trebui să afişeze date dintr-un set de înregistrări, atunci nu va avea nevoie nici de butoane pentru navigare şi nici de selectoare pentru înregistrări. De aceea, dăm proprietăţilor *NavigationButtons* şi *RecordSelectors* valoarea No.
Dacă dorim ca formularul să se comporte ca o cutie de dialog, adică să nu poată fi redimensionat, dăm proprietăţii *BorderStyle* valoarea *Dialog.* Dacă vom vrea ca el să se afle deasupra celorlalte ferestre Access, dăm proprietăţii Popup valoarea Yes. In plus, dacă dorim să se comporte ca o fereastră modală, dăm şi proprietăţii *Modal* valoarea Yes.
Pentru ca utilizatorii să nu îi poată închide de la butonul Close (X) aflat în bara de titlu, dăm şi proprietăţii *CloseButton* valoarea No, iar pentru ca formularul să nu poată fi minimizat sau maximizat, dăm proprietăţii *MinMaxButtons* valoarea *None.*

Crearea unui splash-screen

Pentru a îmbogăţi aplicaţia cu un splash-screen care să stea pe ecran câteva secunde şi să afişeze informaţii despre numele aplicaţiei, versiune, programator etc., sau doar imagine sau un text de bun-venit, vom folosi un formular. Pentru ca el să se comporte ca un spalsh-screen ar trebui ca:

1. Să fie afişat după câteva secunde.
2. Să se închidă după câteva secunde.
3. Să nu poată fi redimensionat direct de către utilizator.
4. Să nu aibă bară de titlu.

In continuare ne vom ocupa, pe rand, de problemele enumerate mai sus.

1. **Pentru ca formularul să fie afişat automat** la deschiderea bazei de date, alegem meniul *Tools*|*Startup*. Se va deschide cutia de dialog Startup. Aici, alegem din caseta combinată *Display Form* numele formularului pe care dorim să-l folosim.
2. **Inchiderea automată a formularului**
   Pentru a face ca formularul să se închidă automat după câteva secunde, vom folosi evenimentul *TimerInterval*. Evenimentul Timer se declaşează prima dată după un anumit interval de la deschiderea formularului şi se repetă regulat la acelaşi interval. Valoarea proprietăţii TimerInterval reprezintă acest interval de timp, în milisecunde. Astfel, dacă dăm proprietăţii TimerInterval valoarea 5000, evenimentul Timer se va declanşa prima oară după cinci secunde de la deschiderea formularului. Nu mai rămâne decât ca în cadrul procedurii de tratare a evenimentului Timer să îi spunem formularului să se închidă. Această procedură va arăta ca mai jos:

```
Private Sub Form_Timer ()
DoCmd.Close
End Sub
```

Notă: Dacă doriţi ca după închiderea ecranului splash să se deschidă un alt formular sau să execute o altă acţiune, tot procedura Form_Timer este locul în care veţi scrie codul necesar acestor acţiuni.

3. **Inchiderea controlată a formularului**
   Pentru ca formularul să nu poată fi redimensionat, dăm proprietăţii *BorderStyle* valoarea Dialog.
   Inchiderea formularului este însă o problemă mai delicată. Deşi el nu va avea o bară de titlu, deci utilizatorii nu vor avea acces nici la butonul Close (X) şi nici la comanda Close din meniul system, ei vor putea totuşi închide formularul cu oricare dintre combinaţiile de taste Ctrl+F4 şi Alt +F4, înainte ca acesta să se închidă automat. Pentru a preveni acest lucru, trebuie să ne asigurăm că formularul nu poate fi închis înainte de producerea evenimentului Timer, care va închide automat formularul. Ori de câte ori utilizatorul încearcă să închidă formularul cu una din combinaţiile de taste amintite anterior, se produce evenimentul Unload, a cărui procedură de tratare are ca argument: Cancel. Dacă acestui argument i se dă valoarea True, formularul nu va mai fi închis. Trebuie deci să ne asigurăm că, atâta timp cât nu s-a produs evenimentul Timer, argumentul Cancel are valoarea True. Pentru aceasta, vom declara la secţiunea (General) (Declarations) pentru modulul formularului variabila booleană bInchide:

```
Dim bInchide As Boolean
```

   La deschiderea formularului, acestei variabile îi dăm valoarea False, scriind următoarea linie de cod în dreptul procedurii pentru tratarea evenimentului Open:

```
Private Sub Form_Open (Cancel As Integer0
bInchide = False
End Sub
```

Abia la producerea evenimentului Timer, variabila bInchide va lua valoarea True.
Deci adăugaţi următoarea linie de cod în cadrul procedurii Form_Timer:

```
bInchide = True
```

Acum, în cadrul procedurii pentru tratarea evenimentului *Unload*, daţi argumentului *Cancel* valoarea *Not bInchide*:

```
Private Sub Form_Unload(Cancel As Integer)
Cancel = Not bInchide
End Sub
```

4. Excluderea barei de titlu

Pagina de proprietăţi a formularului nu conţine nici o proprietate care să specifice prezenţa sau absenţa barei de titlu. Există însă câteva funcţii API care ne vor ajuta în acest scop.
In Windows, caracteristicile unei ferestre sunt specificate de stilul acesteia. Stilul este dat de un întreg de tip Long, deci o valoare pe 32 de biţi, care poate fi privită ca un set de 32 valori binare, fiecare reprezentând o caracteristică a ferestrei. Aceste caracteristici sunt definite prin intermediul unor constante ale căror valori sunt puteri ale lui 2. Pentru a vă asigura că stilul unei ferestre include o anumită caracteristică, trebuie să aplicaţi operţia Or(SAU) pe biţi, între stil şi constanta respectivă. Pentru a renunţa la o caracteristică, aplicaţi operaţia And (ŞI) pe biţi, între stil şi negaţia constantei respective.
Pentru a regăsi stilul, vom apela funcţia API GetWindowsLong. Vom modifica stilul aflat astfel încât să excludă bara de titlu şi apoi vom apela funcţia API SetWindowsLong pentru a-i da ferestrei noul stil.
Funcţia GetWindowsLong primeşte ca argumente identificatorul ferestrei şi constanta GWL_STYLE, care specifică faptul că dorim să returneze stilul ferestrei, iar funcţia SetWindowsLong are ca argumente identificatorul ferestrei, acceaşi constantă.

In acest moment, după ce am modificat stilul ferestrei, formularul se va comporta ciudat dacă nu îi transmitem sistemului Windows să îl redeseneze: bara de titlu va fi în continuare afişată pe ercan, dar Access nu va şti ca ea există. Va trebui să redimensionăm formularul, astfel încât noua dimensiune să fie egală cu dimensiunea formularului fără bara de titlul şi apoi va trebui să mutăm fereastra astfel încât colţul din stânga sus să se găsească pe poziţia de dinaintea îndepărtării barei de titlu.
Vom apela funcţîa API MoveWindows, care redimensionează şi, totodată, repoziţionează fereastra. Aceasta are ca argumente identificatorul ferestrei, coordonatele punctului din stânga-sus, lăţimea şi înălţimea ferestrei. Din păcate însă, problemele noastre nu se sfârşesc atât de simplu. In Windows există două tipuri de coordonate fizice: coordonate ecran, carereprezintă distanţa în pixeli a unui punct faţă de latura din stânga şi, respectiv, de sus a ecanului şi coordinate client, care reprezintă distanţa în pixeli a unui punct faţă de latura de stânga şi, respectiv, de sus a unei ferestre. Funcţia GetWindowsRect returnează coordonatele ecran ale colţurilor din stânga-sus şi, respectiv, dreata-jos ale unei ferestre, coordonate pe care le scrie într-o structură de tip RECT. Pe de altă parte, funcţia MoveWindows aşteaptă ca argumente coordonatele colţului stânga-sus al ferestrei, dar în coordinate client relative la fereastra părinte. Pentru a transforma coordonatele ecran în coordonate client, vom apela funcţia API ScreenClient, care are ca argument identificatorul ferestrei relative la care vor fi date coordonatele client şi punctual ale cărui coordonate vor fi transformate. Fereastra faţă de care trebuie să aflăm coordonatele client ale formularului este fereastra părinte a acestuia (fereastra MDI client). Identificatorul ei îl obţinem apelând funcţia API GetParent, ce are ca argument identificatorul ferestrei copil, adică al formularului nostru.

Introducem la secţiunea (General) (Declarations) a modulului asociat formularului următoarele declaraţii ale celor şapte funcţii API pe care le vom apela, ale celor două tipuri utilizator pe care le vom folosi şi ale constantelor necesare schimbării stilului ferestrei:

```
Private Type RECT
      Left As Long
      Top As Long
      Right As Long
      Booton As Long
End Type
Private Type POINTAPI
      x As Long
      y As Long
End Type
Private Declare Function GetWindowLong Lib ”user32” Alias_
“GetWindowLongA”  (ByVal  hwnd As Long, ByVal  nIndex As Long) As Long
Private Declare Function SetWindowLong Lib “user32” Alias_
“SetWindowLongA”  (ByVal  hnd As Long, ByVal nIndex As Long, byVal_
dwNewLong As Long) As Long
```

```
Private Declare Function GetWindowRect Lib "user32" (ByVal  hwnd As_
Long, IpRect As RECT) As Long
Private Declare Function GetSystemMetrics Lib "user32"  (ByVal nIndex As Long) As Long
Private Declare Function MoveWindows Lib "user32" (ByVal hwnd As_
Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal
bRepaint As Long) As Long
Private Declare Function GetParent Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function ScreenToClient Lib "user32" (ByVal hwnd_
As Long, 1pPoint As POINTAPI) As Long
Const SM_CYCAPTION = 4
Const GWL_STYLE = (-16)
Const WS_CAPTION = &HC00000
```

Modificaţi apoi procedura de tratare a evenimentului Open al formularului, astfel încât ea să efectueze toate operaţiile descrise anterior (deoarece bara de titlu trebuie să fie exclusă la deschiderea formularului, înainte ca acesta să fie ataşat pe ecran):

```
Private Sub Form_Open (Cancel As Integer)
Dim 1Stil As Long
Dim 1StilNou As Long
Dim iInaltimeTitlu As Integer
Dim iLatime As Integer
Dim iInaltime As Integer
Dim rec As RECT
Dim pct As POINTAPI
Dim parent As Long
bInchide = False
'Aflam stilul current al formularului
1Stil = GetWindowLong(Me.hwnd, GWL_STYLE)
'modificam stilul pentru a nu avea bara de titlu
1StilNou -1Stil And Not WS_CAPTION
'inlaturam bara de titlu
Call SetWindowLong(Me.hwnd, GWL_STYLE, 1StilNou)
'aflam dimensiunile ferestrei, cu tot cu bara de titlu
Call GetWindowRect(Me.hwnd, rec)
'aflam inaltimea barei de titlu
iInaltimeTitlu = GetSystemMetrics(SM_CYCAPTION)
'calculam latimea formularului
iLatime = rec.Right – rec.Left
'calculam inaltimea formularului, fără bara de titlu
iInaltime = recBootom – rec.Top – iInaltimeTitlu
parent = GetParent(Me.hwnd)
pct.x = rec.Left
pct.y  = rec.Top
Call ScreenToClient(parent, pct)
'redesenam formularul, cu noile dimensiuni
Call MoveWindows(Me.hwnd, pct.x, pct.y, iLatime, iInaltime, True)
End Sub
```

Crearea butoanelor de navigare personalizate

Atunci când creaţi un nou formular, acesta va avea, în mod implicit, butoane de navigare care să vă ajute să parcurgeţi înregistrările setului pe care se bazează formularul. In plus, tot în partea de jos a formularului, se află şi două casete de text: una afişează numărul înregistrării curente, iar cealaltă, numărul total de înregistrări. Caseta pentru numărul înregistrării curente vă dă şi posibilitatea de a introduce numărul unei înregistrări care să fie afişată, fără a mai fi nevoie să folosiţi butoanele de navigare pentru a ajunge la ea.

Acestea sunt nişte facilităţi importante oferite de Access, dar sunt oare atât de greu de implementat? Să presupunem că nu sunteţi mulţumiţi de designul sau de amplasamentul acestor butoane (în partea de jos a formularului) şi că aţi prefera să lucraţi cu propriile dumneavoastră butoane de navigare. In plus, crearea acestor butoane poate fi un exerciţiu util pentru a învăţa să lucraţi cu înregistrările setului unui formular.

Iată, pe scurt, care sunt problemele pe care trebuie să le avem în vedere:

1. Trecerea corectă de la o înregistrare la alta;
2. Dezactivarea butoanelor de navigare atunci când este cazul;
3. Activarea şi dezactivarea butonului pentru trecerea la o înregistrare nouă;

4. Funcţionarea corectă a casetei de text pentru afişarea înregistrării curente.

Trecerea de la o înregistrare la alta

Să vă reamintim întâi care sunt cele cinci butoane de navigare standard, oferite de Access: |◄ pentru trecerea la înregistrarea următoare, ◄ pentru trecerea la înregistrarea precedentă, ►pentru trecerea la înregistrarea următoare, ►| pentru trecereala ultima înregistrare şi ►* pentru trecerea la o înregistrare nouă. Acestea sunt deci butoanele pe care va trebui să le creăm pentru unul dintre formularele noastre.

Anterior am creat formularul DetaliiProf. Putem lucra în continuare cu el, aşa că dezactivaţi-i butoanele de navigare implicite, dând proprietăţii NavigationButtons valoarea No.

Notă: Deoarece nu ţin de lucrul cu o anumită înregistrare, cel mai potrivit loc de a introduce aceste butoane este secţiunea de subsol a formularului (Form Footer). Pentru a nu apărea neconcordanţe între codul prezentat aici şi controalele create asiguraţi-vă că proprietatea Name a acestora are următoarele valori: cmdPrima pentru butonul Prima, cmdInapoi pentru butonul Inapoi, cmdInainte pentru butonul Inainte, cmdUltima pentru butonul Ultima, cmdAdauga pentru butonul Adaugă, txtCrt pentru caseta de text care afişează numărul înregistrării curente şi txtTotal pentru cea care afişează numărul total de înregistrări.

Codul necesar efectuării acţiunii de trecere la înregistrarea specificată de fiecare buton trebuie să fie, bineînţeles, în cadrul procedurii de tratare a evenimentului Click al butonului resoectiv. Aici, nu trebuie decât să apelăm metoda GoToRecord a obiectului DoCmd. Metodele acestui obiect nu fac altceva decât să ruleze acţiuni (pe care le puteţi include şi într-o macrocomandă). Această metodă are patru argumente, toate opţionale, dintre care cel mai important este al treilea, numit Record, şi care specifică înregistrarea la care se va trece şi deci, care va deveni cea curentă. Valoarea pe care i-o dăm acestui argument poate fi acFirst pentru butonul Prima, acPrevious pentru înapoi, acNext, pentru înainte, acLast pentru Ultima şi acNewRec pentru butonul Adauga.

In continuare, vă prezentăm procedurile pentru tratarea evenimentului Click al celor cinci butoane pe care le-am creat:

```
Private Sub cmdPrima_Click()
DoCmd.GoToRecord , , acFirst
End Sub

Private Sub cmdInapoi_Click()
DoCmd.GoToRecord , , acPrevious
End Sub

Private Sub cmdInainte_Click()
DoCmd.GoRecord , , acNext
End Sub

Private Sub cmdUltima_Click()
DoCmd.GoToRecord , , acLast
End Sub

Private Sub cmdAdauga_Click()
DoCmd.GoToRecord , , acNewRec
```

Dezactivarea controlată a butoanelor

In acest moment, butoanele noastre funcţionează, darn u fără probleme. De exemplu, dacă prima înregistrare este cea curentă, iar noi apăsăm butonul Inapoi sau dacă ultima înregistrare este cea curentă, iar noi apelăm butonul Inainte, va apărea un mesaj de eroare care va spune că nu se poate trece la înregistrarea respectivă.

Iată deci, că în anumite circumstanţe, unele dintre butoane trebuie să fie dezactivate. Pentru a scrie codul care să asigure funcţionarea corectă a butoanelor de navigare, trebuie să stabilim dinainte care va fi comportamentul lor:

1. Dacă prima înregistrare e cea curentă, dezactivăm butoanele Prima şi Inapoi.
2. Dacă ultima înregistrare e cea curentă sau dacă înregistrarea curentă e una nouă, dezactivăm butoanele Inainte şi Ultima.
3. Dacă setul de înregistrări al formularului este de tip Snapshot sau proprietatea Allow Additions a formularului are valoarea No, dezactivăm butonul Adauga.

Bineînţeles, pentru a putea lua aceste decizii, trebuie să ştim în permanenţă care este înregistrarea curentă. De asemenea, trebuie să ştim exact denumirile butoanelor, mai précis valoarea proprietăţii Name a fiecărui buton.

Pentru ca butoanele să funcţioneze corect, introduceţi următoarele linii de cod în cadrul procedurii pentru tratarea evenimentului Current al formularului:

```
Private Sub Form_Current ()
Dim rst As Recordset
```

```
Dim bInregNoua As Boolean
Dim bActualizabil As Boolean
'clonam setul de inregistrari al formularului
Set rst = Me.RecordsetClone
'afisam numarul inregistrarii curente
Me!txtCrt = Me.CurrentRecord
'testam daca inregistrarea curenta e una noua
bInregNoua = Me.NewRecord
'afisam nr. total de inregistrari
If rst.RecordCount <> 0 Then
       Rst.MoveLast
End If
Me!txtxTotal = rst.RecordCount + IIf(bInregNoua, 1, 0)
'aflam daca setul de inregistrari poate fi actualizat
bActualizabil = rst.Updatable And Me.AllowAdditions
Me!cmdAdauga.Enabled = bActualizabil And Not bInregNoua
If bInregNoua Then
       'daca inregistrarea curenta e una noua:
       Me!cmdInainte.Enabled = False
       Me!cmdUltima.Enabled = True
       Me!cmdPrima.Enabled = rst.RecordCount > 0
       Me!cmdInapoi.Enabled = rst.RecordCount > 0
Else
       'astfel, sincronizam clona cu setul de inregistrari
       rst.boockmark = Me.bookmark
       'trecem la înregistrarea precedenta
       rst.MovePrevious
       Me!cmdPrima.Enabled = Not rst.BOF
       Me!cmdInapoi.Enabled = Not rst.BOF
       'ne intoarcem de unde am plecat
       rst.bookmark = Me.bookmark
       'trecem la înregistrarea următoare
       rst.MoveNext
       Me!cmdInainte.Enabled = Not (rst.EOF Or bInregNoua)
       Me!cmdUltima.Enabled = Not (rst.EOF Or bInregNoua)
End If
End Sub
```

Această procedură nu este foarte complicată, dar sunt câteva aspecte pe care le vom detalia. In primul rând, pentru ca să putem naviga printre înregistrările formularului fără ca acest lucru să fie vizibil în formular, folosim proprietatea *RecordsetClone* pentru a obţine o copie a setului de înregistrări, pe care o dăm ca valoare variabilei rst, cu care vom lucra în continuare. Afişăm în caseta de text txtCrt numărul înregistrării curente şi testăm dacă aceasta e o înregistrare nouă. Dacă da, afisăm în caseta de text txtTotal numărul total de înregistrări plus unu. Apoi, după ce am aflat dacă setul poate fi actualizat, activăm butonul cmdAdauga (numai dacă înregistrarea curentă nu e una nouă). Dacă ne aflăm la o înregistrare nouă, dezactivăm butoanele cmdAdauga, cmdInainte şi, dacă setul nu conţine nici o înregistrare, dezactivăm şi butoanele cmdPrima şi cmdInapoi. Altfel, dacă nu suntem la o înregistrare nouă, sincronizăm clona cu setul de înregistrări al forularului, deoarece, după creare, ea nu are o înregistrare curentă şi trecem la înregistrarea precedentă pentru a vedea dacă nu suntem la prima înregistrare. In acest caz, dezactivăm butoanele cmdPrima şi cmdInapoi. Revenim la poziţia iniţială în cadrul clonei şi trecem la înregistrarea următoare. Dacă aceasta e ultima, dezactivăm butoanele cmdInainte şi cmdUltima.

Reactivarea butonului Adauga

In timp ce editaţi o înregistrare nouă, după ce aţi introdus datele veţi dori, poate, să treceţi la o altă înregistrare nouă. Problema este că butonul Adaugă este dezactivat atâta timp cât înregistrarea curentă e una nouă. Trebuie deci să aflăm, cumva, dacă utilizatorul a efectuat vreo modificare asupra noii înregistrări şi dacă da, abia atunci să activăm din nou butonul Adauga. Pentru aceasta, trebuie mai întâi să dăm proprietăţii KeyPreview a formularului valoarea Yes, pentru ca aceasta să recepţioneze mesaje de la tastatură. Astfel, de îndată ce utilizatorul a apăsat o tastă, se va produce evenimentul KeyPress al formularului. Procedura de tratare a acestui eveniment este locul în care vom scrie codul reactivării butonului Adaugă:

```
Private Sub Form_KeyPress (KeyAscii As Integer)
With Me!cmdAdauga
      If Not  .Enabled And Me.Dirty Then
      .Enabled = True
      End If
End With
End Sub
```

Am folosit aici proprietatea Dirty a formularului. Aceasta are valoarea True dacă înregistrarea curentă a fost modificată după ce a fost salvată ultima oară. In acest caz, şi dacă butonul Adauga era dezactivat, îl activăm din nou pentru a putea salva înregistrarea trecând la una nouă.

Caseta de text txtCrt pentru înregistrarea curentă
Pentru a reproduce cât mai fidel funcţionarea mijloacelor de navigare implicite oferite de Access, mai trebuie să-i dăm utilizatorului posibilitetea de a introduce în caseta de text txtCrt numărul care să devină cea curentă, fără a mai fi nevoie să se deplaseze secvenţial până la ea cu ajutorul butoanelor.
Pentru aceasta trebuie ca după ce utilizatorul a introdus o valoare în caseta txtCrt şi aapăsat tasta Enter, noi să regăsim acea valoare. Cu alte cuvinte, va trebui să scriem codul următor în cadrul procedurii pentru tratarea evenimentului AfterUpdate al controlului txtCrt:

```
Private Sub txtCrt_AfterUpdate ()
Dim rst As Recordset
'clonam setul de inregistrari
Set rst = Me.RecordsetClone
On Error GoTotxtCrt_AfterUpdate_Err
'trecem la prima înregistrare
rst.MoveFirst
'trecem la înregistrarea specificată
rst.Move Me!txtCrt - 1
'sincronizăm setul de înregistrări cu clona
Me.bookmark = rst.bookmark
txtCrt_AfterUpdate_Exit:
'închidem clona
rst.close
Exist Sub
'în caz de eroare, afişăm un mesaj şi părăsim procedura
txtCrt_AfterUpdate_Err:
      If rst.RecordCount > 0 Then
      Msg "Introduceţi un număr întreg între 1 şi " &
      rst.RecordCount
End If
Me!txtCrt = Me.CurrentRecord
      Resume txtCrt_AfterUpdate_Exit
End Sub
```

După ce clonăm setul de înregistrări al formularului, trecem la prima înregistrare şi după aceea apelăm metoda *Move*, dându-i ca argument valoarea introdusă de utilizator -1. Cum numerotarea înregistrărilor începe de la 0, pentru a ajunge la înregistrarea dorită trebuie să scădem 1 din valoarea introdusă de utilizator. Metoda *Move* începe deplasarea de la poziţia curentă şi se deplasează cu un număr de înregistrări egal cu argumentul său. De aceea, pentru a ajunge la înregistrarea dorită, a trebuit mai întâi să apelăm funcţia *MoveFirst*, pentru ca prima înregistrare să devină cea curentă.

Atunci când utilizatorul are libertatea de a introduce o valoare într-o casetă de text, trebuie să ne asigurăm că aceasta este rezonabilă. In cazul în care ea va duce la generarea unei erori, trebuie să interceptăm şi să o tratăm corespunzător. De exemplu, dacă utilizatorul introduce o valoare care depăşeşte numărul total de înregistrări sau dacă valoarea nici măcar nu este numerică, afişăm un mesaj care să-i reamintească ce are de făcut, afişăm în txtCrt numărul înregistrării curente, semn că nimic nu s-a schimbat şi părăsim procedura.

Linia de cod
`On Error GoTo txtCrt_AfterUpdate_Err`

interceptează eroarea şi face ca firul execuţiei procedurii să sară direct la linia `txtCrt_AfterUpdate_Err:`

de unde începe tratarea erorilor.

2. Programarea rapoartelor

Access oferă posibilitatea de a perfecţiona rapoartele cu ajutorul codului VBA, prin intermediul proprietăţilor şi evenimentelor acestora. Pentru a pune mai bine în evidenţă legătura ce trebuie să existe între formulare, ce permit introducerea datelor şi rapoarte, al căror scop este prezentarea datelor, vom folosi în acest capitol interogarea DetaliiProfesor.

Reamintim care este instrucţiunea SQL pe care se bazează interogarea DetaliiProfesor.

```
SELECT Profesor.Nume, Profesor.Catedra, Titlu.Titlu,
Titlu.Salariu, Profesor.Statut, Profesor.IdTitlu, Profesor.IdProf
FROM Titlu RIGHT JOIN Profesor ON Titlu.IdTitlu = Profesor.IdTitlu;
```

Creaţi aşadar un nou raport, numit, să zicem, "RapDetaliiProf", cu ajutorul programului ReportWizard. Selectaţi ca sursă de date câmpurile Titlu, Salariu, Nume, Catedra şi Statut ale interogării DetaliiProfesor. In cea de-a doua cutie de dialog a lui ReportWizard alegeţi opţiunea "by Profesor", astfel ca datele să fie grupate în funcţie de profesori, în cea de-a patra alegeţi formatul tabelar (opţiunea Tabular) iar în cea de-a cincea, alegeţi stilul Compact. Lăsăm la latitudinea dumneavoastră chstiunile legate de aspectul raportului; în continuare, noi vom încerca numai să-i îmbunătăţim funcţionalitatea.

Valori booleene

Observaţi faptul că ReportWizard a creat un raport suficient de estetic şi complet pe baza setului de înregistrări ale interogării DetaliiProfesor. Noi, care am proiectat baza de date şi ştim ce semnifică valorile 0 şi 1 ale câmpurilor Statut, ne putem declara mulţumiţi de forma, în care sunt prezentate datele în raport. Gândiţi-vă însă că, de cele mai multe ori, rapoartele ajung în mâinile beneficiarilor dumneavoastră, care nu vor şti întotdeauna să intercepteze astfel de rezultate. De aceea, am putea ca în locul valorilor 0 şi 1 să tipărim chiar semnificaţia lor: "Suplinitor" şi, respectiv, "Titular". Schimbaţi, aşadar, valoarea proprietăţii ControlSource a casetei de text Statut (ataşată iniţial câmpului Statut al setului), scriind în câmpul corespunzător ei următoarea expresie:

```
= IIf([Statut], "Titular", "Suplinitor")
```

O altă posibilitate de a arăta dacă un profesor este titular sau nu, ar fi să înlocuim caseta de text Statut cu o casetă de validare. Aceasta va fi validată dacă profesorul este titular adică câmpul Statut are valoarea 1) şi devalidată, altfel (câmpul Statut are valoarea 0). Stergeţi astfel caseta de text Statut de la secţiunea Detail a raportului şi eticheta sa asociată de la secţiunea Page Header şi, în locul lor, introduceţi o casetă de validare (la secţiunea Detail) şi eticheta acesteia (la secţiunea Page Header). Proprietăţii ControlSource a casetei de validare daţi-i valoarea Statut, iar proprietăţii Caption a etichetei daţi-i valoarea "Titular".

Un efect asemănător aţi fi obţinut şi dacă în locul casetei de validare aţi fi folosit un buton de tip Radio.

Folosirea proprietăţilor rapoartelor

Ca şi formularele, rapoartele au un număr de proprietăţi pe care le puteţi folosi la proiectarea lor pentru a obţine rezultatele dorite.

Niveluri de grupare a datelor

Gruparea şi sortarea datelor este una dintre cele mai importante acţiuni pe care le puteţi efectua asupra unui raport. Este imposibil să separăm complet aceste două probleme, deoarece ele depind una de cealaltă. Secţiunile din cadrul raportului au propriile lor evenimente pentru care puteţi furniza proceduri de tratare şi, mai mult, puteţi adăuga niveluri de grupare şi secţiunile antet/subsol corespunzătoare, prin VBA.

Proprietăţile nivelurilor de grupare

Access administrează nivelurile de grupare ale unui raport cu ajutorul unei matrici, numite GroupLevel, ale cărei elemente sunt indexate începând cu 0. Nu putem accesa un element al acestei matrici direct, ci numai făcând referire la una dintre proprietăţile ei: *GroupFooter* (subsolul grupului), *GroupHeader* (antetul grupului), *GroupInterval* (interval de grupare), *GroupOn* (modul de grupare), *KeepTogether*, *SortOrder* (ordinea de sortare) şi *ControlSource*.
Access nu ne pune la dispoziţie nici un fel de informaţii despre numărul nivelurilor de grupare ale unui raport. Tot ce ştim este că ele pot fi în număr maximum 10 şi că ocupă poziţii consecutive în cadrul matricei GroupLevel. Pentru a număra nivelurile de grupare ale unui raport, trebuie să parcurgem această matrice şi să încercăm să accesăm una dintre proprietăţile sale. Dacă vom obţine o eroare, e clar că acel nivel nu există şi că nici după el nu mai există altele în cadrul matricei. Funcţia *NrNivGr(),* prezentată mai jos, face acest lucru pentru un raport pe care îl primeşte ca argument.

```
Public Function NrNivGr(rap As Report) As Integer
Dim I As Integer
Dim ord As Integer
On Error Resume Next
'iteram printre cele maximum 10 niveluri de grupare
For I = 0 To 9
      ord = rap.GroupLevel(i).SortOrder
```

```
      'in caz de eroare, intrerupem iterarea
      If Err.Number <> 0 Then Exit For
Next i
NrNivGr = i
End Function
```

Pentru a afla numărul nivelurilor de grupare ale raportului RapDetaliiProf, deschidem raportul (fie în modul *Design View*, fie în modul *Preview*) şi scrieţi în fereastra *Immediate*:

```
?NrNivGr(Reports ("RapDetaliiProf"))
```

Cum nu am definit încă nici un nivel de grupare pentru acest raport, rezultatul va fi 0.
In continuare, vom descrie pe scurt proprietăţile unui nivel de grupare.
Proprietăţile *GroupHeader* şi *GroupFooter* au valoarea True (-1) dacă nivelul respectiv are o secţiune antet şi, respectiv, subsol şi False (0), altfel.
Proprietatea *GroupOn* specifică modul e grupare a datelor într-un raport. Valoarea acestei proprietăţi depinde de tipul de date al câmpului sau al expresiei care defineşte nivelul de grupare respectiv. Tabelul următor prezintă valorile posibile ale acestei proprietăţi. Nici una dintre acestea, cu excepţia lui 0 (Each Value) nu are sens dacă nivelul respectiv nu are o secţiune antet sau subsol.

Poziţie	**Descriere**	**Valoare**
Each Value	Fiecare valoare determină un grup	0
Prefix Characters	Inregistrările pentru care valorile câmpului după care se face gruparea încep cu aceleaşi caractere, formează un grup.	
Year	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din acelaşi an, formează un grup.	2
Qtr	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din acelaşi trimestru, formează un grup.	3
Month	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din aceeaşi lună, formează un grup.	4
Week	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din aceeaşi săptămână, formează un grup.	5
Day	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din aceeaşi zi, formează un grup.	6
Hour	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din aceeaşi oră, formează un grup.	7
Minute	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din acelaşi minut, formează un grup.	8
Interval	Când câmpul după care se face gruparea este de tip Date/Time, înregistrările pentru care valorile acestui câmp sunt din acelaşi interval de timp, formează un grup.	9

Proprietatea GroupInterval defineşte un interval pentru valoarea câmpului sau a expresiei care defineşte un nivel de grupare, dacă proprietatea GroupOn nu are valoarea 0 (Each Value). De asemenea, pentru ca această proprietate să poată avea o valoare diferită de 1 (cea implicită), trebuie să existe o secţiune antet sau subsol pentru nivelul de grupare respective. De exemplu, dacă un nivel de grupare e definit de un camp de tip text, iar valoarea proprietăţii GroupOn este 1 (Prefix Characters) şi dacă daţi proprietăţii GroupInterval valoarea 3, fiecare grup al nivelului de grupare va conţine toate înregistrările pentru care primele trei caractere ale valorii câmpului respective sunt aceleaşi. Dacăuparea se face după o coloană de tip Date/Time iar valoarea proprietăţii GroupOn este 5 (Week), valoarea proprietăţii GroupInterval specifică numărul de săptămâni în care se pot încadra valorile câmpului respective pentru înregistrările unui grup.
Proprietatatea KeepTogether specifică dacă, la tipărire, Access va încerca să încadreze datele unui grup pe aceeaşi pagină. Tabelul următor prezintă valorile posibile ale acestei proprietăţi.

Poziţie	Descriere	Valoare
No	Access nu va încerca să tipărească antetul, datele de la secţiunea Detail şi subsolul unui grup pe aceeaşi pagină.	0
Whole Group	Access nu va încerca să tipărească antetul, datele de la secţiunea Detail şi subsolul unui grup pe aceeaşi pagină.	1
With First Detail	Access nu va încerca să tipărească antetul, datele de la secţiunea Detail şi subsolul unui grup pe aceeaşi pagină.	2

Proprietatea ControlSource specifică acea coloană sau expresie care defineşte grupurile din cadrul unui nivel de grupare.
Proprietatea SortOrder precizează ordinea de sortare în cadrul unui nivel de grupare. Valoarea ei poate fi 0, pentru sortarea în sens crescător (0-9 pentru cifre şi A-Z pentru litere) sau -1, pentru sortarea în sens descrescător (9-0 pentru cifre şi Z-A pentru litere).

Lucrul cu nivelurile de grupare

Toate proprietăţile unui nivel de grupare despre care am vorbit până acum pot fi stabilite fie atunci când raportul se află în modul Design View, fie în cadrul procedurii pentru tratarea evenimentului Open al raportului. Astfel, dacă doriţi să oferiţi utilizatorului posibilitatea de a modifica aceste proprietăţi în timp ce raportul se află în modul Preview, va trebui să vă folosiţi de un truc: să treceţi raportul în modul Design View, să efectuaţi modificările cerute şi apoi să-l deschideţi din nou în modul Preview. Pentru ca utilizatorul să nu vadă acest artificiu, daţi proprietăţii (desenare) a raportului valoarea False înainte de a trece în modul Dsign View şi redaţi-i apoi valoarea True după ce aţi redeschis raportul în modul Preview.
Să presupunem că utilizatorul ar fi interesat să poată defini maximum două niveluri de grupare pentru raportul RapDetaiiProf şi apoi să le poată schimba prioritatea. Cele două niveluri de grupare vor fi definite de coloanele Titlu şi Catedra ale interogării DetaliiProf pe care se bazează raportul. Pentru ca utilizatorul să poată specifica aceste niveluri, îi vom pune la dispoziţie tot cutia de dialog Sortare, care, după cum aţi văzut, îi lasă să aleargă atât una sau ambele aceste coloane, cât şi ordinea de sortare corespunzătoare. Pentru a afişa această cutie de dialog în acest scop, vom folosi articolul &Grupare al meniului Acţiuni al barei de meniuri RapDetaliiProfMenu, ataşată raportului. Funcţia asociată acestui articol de meniu se va numi cmdGrupare ().
Iniţial, raportul RapDetaliiProf nu are definit nici un nivel de grupare şi nici nu va putea avea, datorită intervenţiei utilizatorului, decât maximum două. Iată deci logica pe care am ales-o pentru funcţia ataşată comenzii Grupare: dacă raportul nu are nici un nivel de grupare iar utilizatorul a alea una sau ambele coloane în cutia de dialog Sortare, vom crea unul sau, respectiv, două niveluri de grupare, în consecinţă. Dacă raportul are deja un nivel de grupare iar utilizatorul alege o altă coloană în caseta combinată "Primul grup" al cutiei de dialog Sortare, vom modifica proprietatea ControlSource a nivelului respectiv şi anumite controale ale raportului astfel încât ele să reflecte noua grupare. Dacă utilizatorul va alege ambele coloane în cutia de dialog Sortare, vom modifica (dacă este cazul) primul nivel şi vom adăuga unul nou. Dacă raportul are deja definite două niveluri de grupare, utilizatorul nu va mai putea decât să le modifice prioritatea, astfel încât coloana aleasă în prima casetă combinată a cutiei de dialog Source va defini primul nivel de grupare iar coloana rămasă, va defini cel de-al doilea nivel.
Pentru a crea un nivel de grupare şi secţiunile antet/subsol aferente, Access vă pune la dispoziţie funcţia CreateGroupLevel (), care primeşte patru argumente şi returnează indicele nivelului creat:

```
intNiv = CreateGroupLevel (str(NumeRaport,  strExpresie,  bAntet, bSubsol)
```

unde:

strNumeRaport – este un şir de caractere ce conţine numele raportului pentru care creăm noul nivel de grupare;
strExpresie – este un şir de caractere ce reprezintă coloana sau expresia ce defineşte nivelul de grupare;
bAntet şi bSubsol – sunt valori booleene ce arată dacă se vor crea şi secţiunile antet, respectiv subsol, corespunzătoare nivelului de grupare.
Mai jos vă prezentăm funcţia cmdGrupareRap ().

```
Public Function cmdGrupareRap ()
Dim frm As Form
Dim strSort As String
Dim NumeRap As String
Dim iNiveluri As Integer
Numeap = "RapDetaliiProf"
DoCmd.OpenForm "Sortare", WindowMode: =acDialog
If SysCmd(acSysCmdGetObjectState,  acForm,  "Sortare")  Then
      Set frm = Forms Forms ("Sortare")
      'aflăm numărul nivelurilor de grupare existente
      iNiveluri =NrNivGr(Reports(NumeRap))
      With frm
      If IsNull(frm!cmbGr1) Then
      'dacă utilizatorul nu a selectat o coloana în
      'caseta combinata "Primul grup" închidem
      'cutia de dialog şi parasim functia
      DoCmd.close acForm, "Sortare"
      Exit Function
      Else
            If iNiveluri = 2 And IsNull(frm!cmbGr2)  Then
```

```
      'dacă exista doua niveluri de grupare si utilizatorul
      ' nu a selectat nimic în caseta combinata
      ' "Al doilea grup", tiparim un mesaj,
      'inchidem cutia de dialog si parasim functia
         MsgBox "Raportul are doua niveluri de grupare; alegeţi_
         Pentru fiecare coloană si ordinea de sortare!"
        DoCmd.Close acForm, "Sortare"
         Exit Function
      End If
      'dezactivam redesenarea raportului
      Reports(NumeRap).Painting = False
      'deschidem raportul in modul Design View
      DoCmd.OpenRepoart NumeRap, acViewDesign
      If iNiveluri = 0 Then
      'dacă raportul nu are nici un nivel de grupare
      'creăm primul nivel de grupare
         Call CreateGroupLevel(NumeRap,  frm!cmbGr1,  False,  False)
      End If
      'specificam noile valori ale proprietatilor primului nivel
      With Reports(NumeRap).GroupLevel (0)
         .ControlSource = frm!CmbGr1
         .SortOrder = frm!chk1
      End With
      'datele coloanei ce defineste primul nivel de grupare
      'vor fi afisate in prima coloana a raportului, careia
      'îi corespunde eticheta 1b1Gr1 si caseta de text txtGr1
      Reports(NumeRap)!1b1Gr1.Caption = frm!cmbGr1
      Reports(NumeRap)!txtGr1.ControlSource – frm!cmbGr1
     End If
     If Not IsNull(frm!cmbGr2)  Then
      'daca utilizatorul a selectat o coloana în
      'caseta combinata "Al doilea grup":
      'daca am ajuns aici, înseamnă că raportul are
      'cel puţin un nivel de grupare
         If iNiveluri = 1 Then
         'dacă raportul are un sigur nivel de grupare
         'creăm al doilea nivel de grupare
      Call CreateGroupLevel(NumeRap, frm!cmbGr2,  False,  False)
      End If
      'specificam noile valori ale proprietarilor
      'nivelului al 2-lea
      With Reports(NumeRap).GroupLevel (1)
      .ControlSource = frm!cmbGr2
      .SortOrder = frm!chk2
     End  With
      'datele coloanei ce defineste al 20lea nivel de grupare
      'vor fi afisate in a 2-a coloana a raportului, careia
      'îi corespund eticheta 1b1Gr2 si caseta de text txtGr2
      Reports(NumeRap) !1b1Gr2.Caption = frm!cmbGr2
      Reports(NumeRap)!txtGr2.ControlSource = frm!cmbGr2
      End If
End With
DoCmd.OpenReport  NumeRap,  acPreview
Reports(NumeRap).Painting = True
DoCmd.Close  acForm.  "Sortare"
End If
End Function
```

Evenimentele rapoartelor şi ale secţiunilor acestora

Spre deosebire de formulare şi controale, rapoartele au un număr restrâns de evenimente: *Open, Activate, Deactivate, Close, Error, Page* şi *NoData.* Secţiunile unui raport au, la rândul lor, anumite evenimente: Format, Print şi Retreat. In continuare, vom vorbi despre condiţiile în care se declanşează ele şi despre modul în care putem profita de pe urma lor.

Evenimentele rapoartelor

Evenimentul *Open* este generat la deschiderea unui raport în modul *Preview* sau *Print*, înainte ca datele să fie regăsite. De aceea, în procedura de tratare a acestui eveniment, puteţi furniza valori pentru parametrii interogării pe care se bazează raportul (dacă interogarea are parametri) sau chiar pentru a adăuga măsuri de securitate.

Exemplu : Procedură care cere utilizatorului introducerea unei parole înainte ca raportul să fie deschis

```
Private Sub Report_Open(Cancel As Integer)
Dim strParola = InputBox("Introduceţi parola", "Parola")
In strParola <> "Raport, deschide-te!"
   Cancel = True
End If
End Sub
```

Dacă parola furnizată de utilizator nu este "Raport, deschide-te!", argumentul Cancel ia valoarea True ceea ce face ca operaţia de deschidere a raportului să nu mai aibă loc.
Evenimentul Activate este generat atunci când un raport care este deja deschis primeşte focusul şi deci devine fereastra activă. In procedura de tratare a acestui eveniment puteţi afişa, de exemplu, o bară de instrumente.

```
Private Sub Report_Activate ()
DoCmd.ShowToolbar "toolbar"
End Sub
```

unde "toolbar" este numele unei bare de instrumente creată de dumneavoastră (cu ajutorul metodei Show Toolbar a obiectului DoCmd puteţi afişa şi barele de instrumente predefinite).

Evenimentul Deactivate se declanşează atunci când raportul pierde focusul în favoarea altei ferestre din Access. In procedura pentru tratarea acestui eveniment puteţi ascunde bara cu instrumente afişată la activare:

```
Private Sub Report_Activate()
DoCmd.ShowToolbar 'toolbar",  acToolbarNo
End Sub
```

Evenimentul Close apare la închiderea unui raport. El poate fi folosit pentru afişarea unui mesaj, pentru înregistrarea diferitelor informaţii etc.
Evenimentul Error este declanşat când motorul Jet Engine generează o eroare (de exemplu, dacă raportul se bazează pe un set de înregistrări care nu există sau dacă una dintre tabelele de bază este blocată de alt utilizator). Procedura de tratare a acestui eveniment are două argumente: DataErr care reprezintă numărul asociat erorii şi Response care specifică dacă Access va afişa sau nu un mesaj de eroare. Dacă daţi argumentului Response ca valoare constantă intrinsecă acDataContinue, Access va ignora eroarea şi nu va afişa mesajul iar dacă îi daţi valoarea acDataErrDisplay, Access va afişa standard de eroare.
Următoarea procedură de tratare a acestui eveniment verifică dacă raportul se bazează pe un set de înregistrări care nu există (adică valoarea lui DataErr este 2580), caz în care afişează un mesaj personalizat iar altfel, afişează mesajul standard.

```
Private Sub Report_Error (DataErr As Integer, Response As Integer)
If DataErr = 2580 Then
      MsgBox "Tabela sau interogarea:  " & Me.RecordSource  & _
      "  pe care se bazează raportul nu există", vbExclatation
   Response = acDateErrContinue
Else
   Response  = acDataErrDisplay
End If
End Sub
```

Evenimentul Page este generat după ce o pagină a raportului a fost formatată, dar înainte ca aceasta să fie tipărită. In cadrul procedurii sale de tratare, puteţi adăuga diferite efecte grafice ce ţin de întreaga pagină şi nu doar de una dintre secţiunile raportului. De exemplu, puteţi desena un dreptunghi care să încadreze întreaga pagină, folosind metoda Line:

```
Private Sub Report_Page()
Me.Line (0, 0) -(Me.ScaleWidth, Me.ScaleHeight), , B
End Sub
```

Metoda Line a unui obiect de tip raport desenează drepte şi dreptunghiuri pe raportul respectiv. Prima pereche de coordonate, (0,0), reprezintă punctul de la care începe desenarea, iar cea de-a doua, (Me.ScaleWidth, Me.ScaleHeight), punctul de sfârşit. Argumentul B indică faptul că se va desena un dreptunghi şi nu o dreaptă. Proprietăţile ScaleWidth şi ScaleHeight ale unui raport reprezintă lăţimea şi, respectiv, înălţimea acestuia. Unitatea de măsură implicită în Access se numeşte twip şi este 1/567 dintr-un centimetru. Pentru a modifica unitatea de măsură pentru coordonatele punctelor unei pagini, folosiţi proprietatea ScaleMode a raportului.

Evenimentul *NoData* este generat atunci când sursa de date a raportului nu conţine nici o înregistrare. Această situaţie poate apărea frecvent în urma filtrării excesive a unui raport. Procedura de tratare a acestui eveniment are un singur argument, Cancel, căruia dacă îi daţi valoarea True, raportul nu va mai fi deschis iar dacă îî daţi valoarea False, raportul va fi deschis dar nu va afişa nici o înregistrare. Puteţi folosi acest eveniment pentru a afişa un mesaj:

```
Private Sub Report_NoData(Cancel,  As Integer)
MsgBox "Nu am ce afişa!"
End Sub
```

Evenimentele secţiunilor unui raport

Evenimentele secţiunilor unui raport sunt legate de formatarea şi tipărirea datelor acestuia: Format, Print şi Retreat.
Evenimentul Format este generat după ce Access a regăsit datele ce trebuie să fie afişate în cadrul secţiunii respective, dar înainte ca acestea să fie afişate sau tipărite. Astfel, aveţi posibilitatea să aduceţi raportului modificări de ultimă oră (de exemplu, să faceţi un control vizibil sau invizibil) sau să calculaţi diverse totaluri. Procedura sa de tratare are două argumente: Cancel şi FormatCount. Dacă daţi argumentului Cancel valoarea True, secţiunea respectivă nu va mai fi formatată şi se va trece la următoarea secţiune. Cel de-al doilea argument, FormatCount, indică de câte ori a fost formatată secţiunea respectivă. De exemplu, să presupunem că secţiunea Detail a unui raport ocupă două linii pe pagină iar proprietatea KeepTogther a acestei secţiuni are valoarea Yes. Dacă pe o pagină nu încape decât prima linie, ambele linii vor fi tipărite pe pagina următoare şi deci secţiunea va trebui să fie formatată din nou. In acest caz, valoarea argumentului FormatCount va fi 2.

Proprietatea FormatCount a unei secţiuni

Proprietatea FormatCount specifică de câte ori a fost formatată o secţiune. Dacă dorim să numărăţi secţiunile tipărite, puteţi face acest lucru incrementând valoarea unei variabile în cadrul procedurii de tratare a evenimentului Format. Dar, acest eveniment se poate apela şi de două ori pentru aceeaşi secţiune iar dumneavoastră puteţi să numărăţi secţiunea o singură dată. Puteţi face acest lucru incrementând valoarea variabilei numai dacă proprietatea FormatCount are valoarea 1.
Pentru a ilustra lucrul cu evenimentele secţiunile unui raport, vom reveni la raportul Student_Cursuri. Pentru acest raport am definit doă niveluri de grupare a datelor: primul, în funcţie de coloana Grupa a interogării Curs_Studenţi ce stă la baza raportului, are o secţiune antet, iar al doilea, în funcţie de coloana NrMatricol, are atât o secţiune antet, cât şi una subsol. Vă reamintim care este interacţiunea SQL a interogării Curs_Studenţi:

```
SELECT Curs.Denumire, Curs_Student.Nota, Student.NumeSt,
Student.PrenumeSt, Student.Grupa, Student.NrMatricol
FROM Student  INNER  JOIN  (Curs INNER  JOIN  Curs_Student  ON
  (Curs.IdCurs  = Curs_Student.IdCurs)  AND
  (Curs.IdCurs  = Curs_Student.IdCurs)  AND
  (Curs.IdCurs  = Curs_Student.IdCurs) )
ON Student.NrMatricol = Curs_Student.NrMatricol;
```

Să presupunem acum că ne interesează numai primele două cursuri la care un student a obţinut cele mai mari note dintre toate la care s-a înscris şi media pentru acestea două. Pentru aceasta, nu trebuie decât să sortăm raportul în ordinea descrescătoare a valorilor coloanei Nota şi apoi să tipărim primele maximum două cursuri (pot fi şi studenţi care s-au înscris la un sigur curs). Sortarea o puteţi face cu ajutorul cutiei de dialog Sorting and Grouping, specificând ordinea descrescătoare pentru coloana Nota. Pentru a putea afişa maximum două cursuri şi a calcula media notelor obţinute numai la acestea, va trebui să folosim două variabile globale în cadrul modulului raportului. Adăugaţi deci la secţiunea (General) (Declarations) a modulului următoarele două declaraţii:

```
Private NrCursuri As Integer
Private SumaNote As Integer
```

Pentru fiecare student în parte va trebui să iniţializăm valoarea acestor două variabile. Putem face acest lucru în cadrul procedurii de tratare a evenimentului Format al antetului celui de-al doilea nivel de grupare:

```
Private Sub GroupHeader2_Format(Cancel) As Integer, FormatCount _
As Integer)
NrCursuri = 0
SumaNote = 0
End Sub
```

Apoi, pentru fiecare curs în parte (deci pentru fiecare înregistrare de la secţiunea Detail) vom incrementa aceste valori numai dacă numărul cursurilor nu este > 2. pentru aceasta, vom folosi evenimentul Format al secţiunii Detail:

```
Private Sub Detail_Format(Cancel As Integer, FormatCount As
Integer)
If FormatCount = 1 Then
      If NrCursuri < 2 Then
      NrCursuri = NrCursuri + 1
      SumaNote = SumaNote + [Nota]
      Else: Cancel = True
      End If
End If
End Sub
```

Media va fi afişată în caseta de text txtMedia de la secţiunea subsol a nivelului al doilea de grupare (corespunzătoare fiecărui student). Cunoaştem atât suma celor mai mari maximum două note obţinute, cât şi numărul cursurilor (unul sau două), deci nu mai trebuie decât ca, în cadrul procedurii de tratare a evenimentului Format al secţiunii subsol a nivelului de grupare definit de coloana NrMatricol, să dăm casetei txtMedia ca valoare câtul lor:

```
Private Sub GroupFooter3_Format (Cancel As Integer,  FormatCount As
Integer)
txtMedia = SumaNote / NrCursuri
End Sub
```

Proprietăţi ale secţiunilor disponibile numai la rulare

In plus, faţă de proprietăţile accesibile în modul Design View prin intermediul paginii de proprietăţi, secţiunile unui raport mai au şi proprietăţi disponibile numai în timpul desfăşurării evenimentelor legate de acestea (formatare, tipărire). Una dintre aceste proprietăţi este şi FormatCount, despre care am vorbit la secţiunea anterioară. In continuare, ne vom ocupa de celelalte proprietăţi de acet tip şi de utilizarea lor.

Proprietatea MoveLayout, dacă are valoarea True, face ca Access să treacă la următoarea locaţie pentru tipărire de pagină (i.e. secţiunea este deplasată la poziţia următoare). Dacă are valoarea False, nu se efctuează deplasarea. Poate fi folosită pentru inserarea de spaţii libere într-un raport, după cum veţi vedea din exemplul următor.

Proprietatea NextRecord, dacă are valoarea True, face să se treacă la următoarea înregistrare pentru secţiunea respectivă. Dacă are valoarea False, rămâne la înregistrarea curentă.

Proprietatea PrintScreen specifică dacă secţiunea va fi sau nu tipărită (dacă are valoarea True, secţiunea va fi tipărită, dacă are valoarea False, nu).

Proprietatea FormatCount indică de câte ori a fost generat evenimentul Format pentru o secţiune.

Proprietatea PrintCount indică de câte ori a fost generat evenimentul Print pentru o secţiune.

Proprietatea WillContinue este accesibilă din cadrul procedurii de tratare a evenimentului Print şi are valoarea True dacă se va continua tipărirea secţiunii curente pe pagina următoare.

Proprietatea HasContinued este accesibilă din cadrul procedurii de tratare a evenimentului Format şi are valoarea True dacă tipărirea secţiunii curente s-a continuat de pe pagina anterioară (adică, dacă valoarea proprietăţii FormatCount pentru secţiunea respectivă este 1).

Vom încheia acest capitol cu un exemplu care să folosească unele dintre proprietăţile de mai sus, dar şi cutia de dialog Page Setup, pentru a obţine o listă nominală pe două coloane a studenţilor din baza de date şi a grupelor acestora. Raportul, pe care l-am numit “ListaStudenti”, se bazează pe următoarea instrucţiune SQL:

```
SELECT (NumeSt) & " " &[PrenumeSt] As Nume,  Student,  Grupa
FROM Student;
```

Puteţi fie să salvaţi interogarea şi să folosiţi ReportWizard pentru a crea raportul, fie să-l creaţi dumneavoastră, pornind de la zero, în modul Design View.

Datele raportului sunt grupate după prima literă a numelor studenţilor şi sunt sortate alfabetic. Adăugaţi deci, cu ajutorul ferestrei Sorting and Grouping, un nivel de grupare pe baza coloanei Nume a interogării (ce reprezintă numele complet al studentului

obţinut prin concatenarea numelui şi prenumelui), specificaţi existenţa unui antet şi daţi proprietăţii GroupOn valoarea Prefix Characteristics iar proprietăţii GroupInterval, valoarea 1.
La secţiunea antet a nivelului de grupare adăugaţi o casetă de text, care va afişa prima literă a valorii câmpului Nume pentru un grup, deci proprietatea sa ControlSource, va avea valoarea:

```
= Left ([Nume],  1)
```

Dacă vom trece acum în modul Preview, vom constata că datele ocupă o suprafaţă mică din pagină, care nu este deci folosită eficient. Putem face ca datele să fie tipărite pe mai multe coloane, pentru a economisi hârtia şi a avea o imagine mai cuprinzătoare asupra lor. Pentru aceasta, selectaţi comanda File | Page Setup şi, în cutia de dialog *Page Setup* ce se va deschide, alegeţi pagina Columns. Aici, introduceţi în câmpul *Number of Columns* (numărul de coloane) valoarea 2, în câmpul *Width* (lăţimea unei coloane) valoarea 3 (aproximativ 7,5 cm pentru fiecare coloană), iar în grupul *Column Layout*, alegeţi opţiunea *Down then Across* (ce specifică, aşa cum indică şi săgeţile, faptul că întâi va fi completată cu date o coloană şi apoi se va trece la cealaltă).

De multe ori, atunci când priveşte o astfel de listă cu foarte multe nume, utilizatorul nu poate să o urmărească atent. Pentru a rezolva această problemă, se obişnuieşte inserarea unui spaţiu la fiecare grup câte 3-5 linii. Vom face şi noi acest lucru pentru fiecare grup al raportului. Si de această dată, vom folosi o variabilă globală a modulului asociat raportului, a cărei declaraţie este:

```
Dim iCont As Integer
```

Pentru a adduce la 0 valoarea acestei variabile pentru fiecare grup, folosiţi procedura de tratare a evenimentului Format al secţiunii antet a nivelului de grupare:

```
Private Sub GroupHeader0_Format (Cancel As Integer,  FormatCount As Integer)
iCount = 0
End Sub
```

In cadrul procedurii de tratare a evenimentului Format al secţiunii Detail, vom avea grijă că după fiecare trei înregistrări să inserăm o linie liberă:

```
Private Sub Detail_Format (Cancel As Integer,  FormatCount  As Integer)
If FormatCount = 1 Then
      If iCont = 3 Then
      Me.MoveLayout = True
      Me.PrintSection = False
      Me.NextRecord = False
      iCont = 1
   Else iCont = iCont + 1
   End If
End If
End Sub
```

Observaţi că am folosite proprietăţile *MoveLayout, PrintSection* şi *NextRecord*. Astfel, am indicat faptul că dacă s-a tipărit un grup de trei înregistrări, trecem la următoarea poziţie de pagină (Me.MoveLayout = True), nu tipărim înregistrarea curentă (Me.PrintSection = False) şi nici nu trecem la înregistrarea următoare (Me.NextRecord = False). Astfel, listăm, pur şi simplu, o linie liberă, fără a omite nici o înregistrare.