Curs 10. Lucrul cu obiecte

1. **Obiectele din Access**
2. **Lucrul cu variabile de tip obiect**
3. **Colecţii implicite**
4. **Lucrul cu proprietăţile obiectelor**

Unul dintre motivele principale ale apariţiei programării orientate pe obiecte este nevoia de a modela cât mai bine fenomenele şi entităţile din lumea înconjurătoare. Caracteristica cea mai importantă a obiectelor este aceea că ele conţin toate informaţiile necesare pentru a funcţiona. Astfel, întâlnim obiecte în toate domeniile programării: programare obiectuală sistem, sisteme de gestiune a bazelor de date orientate obiect, programare obiectuală în proiectare, jocuri orientate obiectual. Toate acestea se datorează avantajelor pe care aceasta le oferă faţă de limbajele procedurale tradiţionale:

- eficienţă mai mare datorată posibilităţii de a refolosi cantităţi însemnate de cod din alte aplicaţii, de unde rezultă şi creşterea complexităţii şi calităţii aplicaţiilor;
- posibilitatea de a accesa date stocate pe platforme şi sisteme diferite;
- posibilitatea de a oferi utilizatorilor metode grafice intuitive şi prietenoase de comunicare cu sistemul;

In continuare, vom explica termenii cheie ce apar atunci când vorbim despre obiecte.

Proprietăţi

Toate obiectele au proprietăţi ce le caracterizează. *Exemplu:* În Access un formular are şi el o mulţime de proprietăţi, cum ar fi: titlul (Caption), filtrul (Filter) etc. Proprietăţile unui obiect pot fi modificate fie cu ajutorul paginii de proprietăţi în modul *Design View*, fie prin intermediul limbajului VBA.

Metode

Metodele reprezintă modul în care un obiect efectuează diferite acţiuni. *Exemplu:* În Access obiectul bază de date (*Database*) are o metodă pentru a crea un nou set de înregistrări (*OpenRecordset*).

Clase

O clasă reprezintă mulţimea tuturor obiectelor care au aceleaşi proprietăţi şi metode. Un obiect al unei clase poartă numele de instanţă a clasei respective. O clasă poate avea mai multe subclase (sau clase derivate) care moştenesc proprietăţile şi metodele clasei de bază dar, pe lângă aceasta, mai au şi propriile lor proprietăţi şi metode, ce le particularizează.

1. Obiectele din Access

In Access există două seturi de obiecte: obiectele Access generale (prezentate în tabelul de mai jos) şi obiecte pentru accesarea datelor (*Data Access Objects* sau, pe scurt, DAO):

Obiect Access	Descriere
Application	Obiectul Microsoft Access
Control	Un control de pe un formular sau raport
DoCmd	Acţiuni în VBA
Immediate	Fereastra Immediate
Form	Un formular sau subformular deschis
Module	Un modul
Report	Un raport sau subraport decshis
Screen	Ecranul
Section	O secţiune a unui formular sau raport
Pages	Paginile unui control
References	Referinţe la alte biblioteci de obiecte

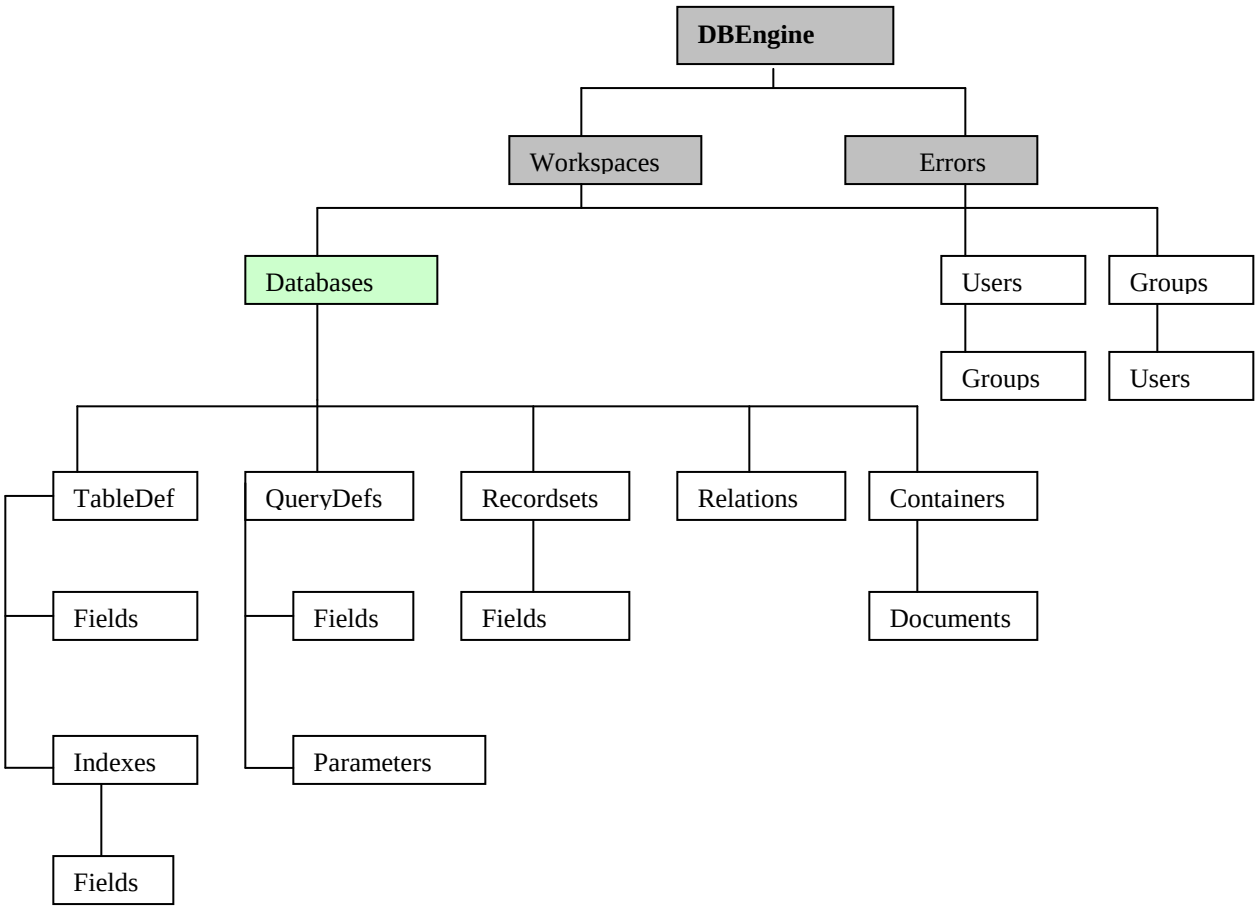
Obiect DAO	Descriere
Container	Un obiect care conţine informaţii despre alte obiecte
Database	O bază de date deschisă
DBEngine	Motorul bazei de date
Document	Informaţii pe care motorul bazei de date le administrează, despre alte obiecte
Error	Errori generate de accesarea datelor
Group	Un cont al unui grup de utilizatori
Index	Un index pe o tabelă
Parameter	Un parametru al unei interogări

Property	Proprietatea unui obiect
QueryDef	O interogare salvată
Recordset	Un set de înregistrări definit de o tabelă sau interogare
Relation	O relaţie între două tabele sau interogări
TableDef	O tabelă salvată
User	Contul unui utilizator
Workspace	O sesiune de lucru cu o bază de date

In continuare, vom vorbi despre obiectivele DAO şi despre cum se folosesc ele.

Colecţii

O bază de date este un obiect care conţine, la rândul său, alte obiecte. In mod asemănător, formularele conţin controale, iar tabelele, câmpuri. Obiectele care sunt conţinute într-un alt obiect sunt grupate în colecţii. Colecţiile pot fi formate numai din obiectele aceleiaşi clase. Astfel, dacă un obiect conţine mai multe formulare, acestea vor forma colecţia *Forms* (formulare) a obiectului respectiv. O colecţie poate fi deci considerată ca fiind un vector.
Figura următoare prezintă obiectele DAO sub forma unei diagrame ierarhice ce arată legăturile dintre obiecte şi colecţiile lor.



In vârful ierarhiei se află obiectul DBEngine (motorul bazei de date). El conţine două colecţii: *Errors* şi *Workspaces* (erori şi sesiuni de lucru). Un Workspace este ca un birou al unei clădiri, clădirea fiind motorul *DBEngine*. Colecţia *Errors* conţine o listă cu erorile apărute în DBEngine.
Aproape toate acţiunile utilizatorului sunt procesate de motorul DBEngine şi sunt efectuate asupra unui obiect DAO. De aceea, este foarte important să înţelegem ce sunt obiectele pentru accesarea datelor şi care sunt relaţiile dintre ele. Pentru aceasta, vom folosi fereastra *Immediate* care ne va ajuta să vedem obiectele stocate în baza de date şi cum sunt organizate în ierarhia DAO.

Exemple:
1. Creăm un nou modul standard în baza de date curentă (CursuriOptionale.mdb).
   Deschidem fereastra *Immediate* şi scriem:

```
?DBEngine.Workspaces(0).Name
```

După ce apăsăm tasta Enter, vom primi următorul rezultat:

```
# Default Workspace#
```
(adică sesiunea de lucru implicită).

Prin linia de cod scrisă mai sus am cerut numele unei sesiuni de lucru. Obiectul DBEngine, care se află în vârful ierarhiei DAO, conţine două colecţii: Workspaces şi Errors. Colecţia *Workspaces* este, de fapt, o matrice ale cărei elemente sunt obiecte de tip Workspace. Astfel, DBEngine.Workspaces(0).Name înseamnă "proprietatea Name (nume) a elementului cu indicele 0 al colecţiei Workspaces a obiectului DBEngine".

2. Să coborâm pe o treaptă inferioară ierarhiei, cerând numele bazei de date curente. Pentru aceasta, introducem în fereastra *Immediate*:

```
?DBEngine.Workspaces(0).Databases(0).Name
```

Rezultatul va fi calea completă şi numele fişierului în care se află baza de date curentă.
Fiecare obiect de tip Workspace conţine trei colecţii: *Users* (utilizatori), *Groups* (grupuri) şi *Databases* (baze de date) care, la rândul lor, pot fi privite ca matrice (vectori). Prin linia de cod de mai sus am cerut numele obiectului de tip Database cu indicele 0 din cadrul colecţiei Database a obiectului Workspaces(0) al obiectului DBEngine.

3. Mai coborâm o treaptă a ierarhiei şi cerem numele unui obiect al unei colecţii a obiectului, DBEngine.Workspaces(0).Databases.(0)

```
?DBEngine.Workspaces(0).Databases(0).TableDefs(0).Name
```

TableDefs(0) e obiectul cu indicele 0 din matricea TableDefs (tabele salvate).
Rezultatul va fi numele unei tabele din baza de date curentă, şi anume, Curs. Dacă înlocuim TableDefs(0) cu TableDefs(3), vom obţine un rezultat mai neobişnuit: MSysACEs. Aceasta deoarece tabelele sunt stocate în ordine alfabetică în cadrul matricei TableDefs, astfel încât tabelei sistem MSysACEs (folosită de Access) îi corespunde indicele 3 (primele trei tabele sunt, în ordine alfabetică: Curs, Curs_Prof şi Curs_Student).

Observăm că între obiectele de pe trepte consecutive ale ierarhiei şi între obiecte şi proprietăţile lor se pune un punct. Continuând ca mai sus, putem găsi informaţii despre toate obiectele stocate într-o bază de date.

Notă: E important să sesizăm diferenţa dintre o colecţie şi tipul elementelor ce o compun. De exemplu, Workspaces (la plural) este numele colecţiei, iar Workspace (la singular) este tipul elementelor pe care ea le conţine.

Obiectul DBEngine (motorul Jet Engine)

Este obiectul aflat în vârful ierarhiei DAO, este un obiect predefinit ce nu poate fi creat. Există un singur obiect DBEngine pentru o aplicaţie. El nu face parte din nici o colecţie şi conţine toate celelalte obiecte. El poate fi folosit pentru a compacta sau repara o bază de date, pentru a înregistra baze de date ODBC, pentru a obţine versiunea motorului Jet şi pentru a stabili timpul maxim necesar deschiderii sesiunii unui utilizator.

Colecţia Errors (Erori)

Un obiect de tip Error primeşte toate erorile apărute în urma eşuării unei acţiuni efectuate asupra unui obiect DAO. Erorile sunt ordonate în cadrul colecţiei în funcţie de codul lor.

Colecţia Workspaces (Sesiuni de lucru)

Un obiect de tip Workspace defineşte o sesiune de lucru pentru un utilizator. Acest obiect conţine toate bazele de date deschise de acel utilizator. Tranzacţiile efectuate în cadrul unei sesiuni (obiect de tip Workspace) sunt globale tuturor bazelor de date ale sesiunii respective. Access creează implicit obiectul Workspaces (0). Dacă nu au fost impuse măsuri de securitate asupra bazei de date curente, proprietăţii Name (nume) a acestui obiect i se dă valoarea #Default Workspace#, iar proprietăţii UserName (numele utilizatorului) i se dă valoarea Admin (administrator). Acest obiect implicit nu poate fi şters sau închis fiind disponibil tot timpul.

Colecţia Databases (Baze de date)

Un obiect de tip Database reprezintă o bază de date deschisă sau creată cu DAO. Cu ajutorul metodei *Create Database* a unui obiect Workspace, se adaugă automat o bază de date la colecţia Databases a sesiunii respective. O bază de date poate fi închisă cu

ajutorul metodei Close, ceea ce o va şi şterge din cadrul colecţiei. Colecţia DataBases conţine toate bazele de date deschise cu DAO, plus baza de date curentă, deschisă în Access (care este obiectul DBEngine Workspaces(0).Databases(0) sau CurrentDB).

Colecţia User (Utilizatori)

Un obiect de tip User reprezintă contul unui utilizator.

Colecţia Groups (Grupuri)

Fiecare interogare salvată în Access sau creată cu ajutorul metodei CreateQueryDef este reprezentată printr-un obiect de tip QueryDefs. Obiectele de tipQueryDef sunt deci instrucţiuni SQL precompilate. Cu ajutorul unui obiect de tip QueryDef puteţi crea seturi de înregistrări, îi puteţi regăsi instrucţiunea SQL, puteţi afla dacă returnează sau nu înregistrări sau puteţi executa interogarea.

Colecţia TableDef (Tabele)

Un obiect de tip Table Def reprezintă o tabelă stocată într-o bază de date. Tabela se poate afla în baza de date curentă sau poate fi ataşată dintr-o bază de date externă. Cu ajutorul unui obiect de tip TableDef puteţi afla dacă tabela e ataşată, îi puteţi afla regulile de validare dacă poate fi sau nu actualizată sau numărul de înregistrări.

Colecţia Indexes (Indecşi)

Un obiect de tip index reprezintă un index al unei tabele sau al unui set de înregistrări.

Colecţia Fields (Câmpuri)

Un obiect de tip Field reprezintă o coloană. Obiectele de tip Relation (relaţie), Recordset, TableDef şi Index conţin câte o colecţie Fields. Atributele unui camp poate fi aflate şi modificate prin intermediul proprietăţilor obiectului Field corespunzător.

Colecţia Recordsets (Seturi de înregistrări)

Obiectele de tip Recordset (set de înregistrări) sunt cel mai des folosite şi deci, poate cele mai importante obiecte DAO. Aceste obiecte sunt temporare (nu sunt salvate pe disc).

Colecţia Relations (Relaţii)

Fiecare relaţie dintre două sau mai multe tabele ale unei baze de date Access e reprezentată printr-un obiect de tip Relation. Colecţia Relations a unui obiect de tip Database conţine toate relaţiile definite în baza de date respectivă.

Colecţia Parameters (Parametri)

Parametrii declaraţi cu ajutorul cuvântului cheie PARAMETERS în instrucţiunea SQL a unei interogări se numesc parametrii formali. Colecţia Parameters a unui obiect de tip QueryDef e formată din toţi parametrii formali definiţi pentru interogarea respectivă. Nu putem şterge sau adăuga obiecte la colecţia Parameters.

Colecţia Properties (Proprietăţi)

Un obiect de tip Property reprezintă o caracteristică a unui obiect. Fiecare obiect DAO are o colecţie Properties. Un obiect de tip Property poate fi o proprietate predefinită sau una definită de utilizator. Programatorul poate adăuga proprietăţi (cu ajutorul metodei CreateProperty) unui anumit obiect şi este responsabil pentru stabilirea şi modificarea valorilor acestor proprietăţi. Proprietăţile definite de utilizator sunt singurele care pot fi şterse din colecţia Properties a unui obiect; proprietăţile predefinite nu pot fi şterse.

Containere şi documente

Un container este o colecţie de obiecte salvate în Access: baze de date, formulare, rapoarte, module, relaţii, tabele. Termenul "document" este o descriere generică pentru un obiect stocat într-un container. Pentru a înţelege mai bine ce reprezintă containerele şi documentele, gândiţivă la fereastra Database. Fiecare pagină a acestei ferestre reprezintă un container, iar fiecare obiect dintr-o pagină este un document.

Exemplu: Subrutină pentru a examina containerele şi documentele bazei de date CursuriOptionale.mdb.

```
Sub Containere_Documente( )
Dim dbCrt As Database
Dim conCrt As Container
Dim docCrt As Document
Set dbCrt = DBEngine.Workspaces(0).Databases(0)
```

```
For Each conCrt In dbCrt.Containers
      Debug.Print "Container:" & conCrt.Name
      For Each docCrt In conCrt.Documents
      Debug.Print "Document: " & docCrt.Name
      Next
Next
End Sub
```

Am definit trei variabile: una de tip Database, care indică baza de date curentă, una de tip container care indică fiecare element din colecţia Containers şi una de tip Document, ce indică fiecare document dintr-un container.

Notă: Observăm că pentru a atribui variabilei dbCrt baza de date curentă am folosit cuvântul cheie Set, pe care trebuie sa-l folosim întotdeauna pentru a atribui valori obiectelor.
Pentru a itera printre elementele colecţiilor Containers, am folosit instrucţiunea *For... Each*, ce se aseamănă cu instrucţiunea *For…Next*, cu deosebirea că este creată special pentru parcurgerea colecţiilor.

2. Lucrul cu variabile de tip obiect

Când declarăm o variabilă obişnuită, îi cerem lui Access să aloce spaţiul necesar stocării unei informaţii pentru care am specificat un tip de date (dacă nu specificaţi nici un tip de date, Access va considera în mod implicit că tipul e variant). Lucrurile sunt puţin diferite la declararea unei variabile de tip obiect. In acest caz, Access creează numai un pointer la un obiect.
Exemplu: nici una dintre declaraţiile de mai jos:

```
Dim db Database
Dim form As Form
Dim ct1 As Control
```

nu stochează informaţii şi nu indică vreun obiect existent. Pentru a le face să indice un obiect, trebuie să folosim cuvântul cheie *Set*, ca mai jos:

```
Set dbCrt = DBEngine.Workspaces(0).Databases(0)
Set form = db.Forms!DetaliiProf
Set ct1  = frm.Controls!Nume
```

Legătura dintre variabile şi obiectul spre care acesta indică este distrusă (şi memoria necesară este eliberată) o dată cu expirarea duratei de viaţă a variabilei. Dacă dorim să facem explicit, acest lucru, vom folosi valoarea predefinită Nothing:

```
Set frm = Nothing
```

După cum am mai văzut şi în exemplele anterioare, pentru a accesa un anumit obiect trebuie să parcurgem ierarhia în sens descrescător, până la acel obiect.

```
DBEngine.ColectieParinte.ColectieCopil("ObiectCopil")
```

Tabelul următor prezintă cele patru metode prin care putem accesa un element al unei colecţii:

Sintaxa	Exemplu	Explicaţii
colecţie ("nume")	DBEngine.Workspaces(0). Databases_("Cursuri Optionale")	
colectie (var)	strBazaDate = "Cursuri Optionale" DBEngine.Workspaces(0).Databases_ (strBazaDate)	var e o variabilă de tip şir de caractere sau Variant
colectie (indice)	DBEngine.Workspaces(0).Databases(0).	indice reprezintă poziţia obiectului în cadrul colecţiei
colectie!nume sau colectie![nume]	DBEngine.Workspaces(0).Databases!_ [Cursuri Optionale]	Parantezele drepte sunt necesare numai dacă numele colecţiei conţine caractere speciale (de exemplu, spaţii)

Astfel, dacă vrem să acceptăm un obiect al cărui nume îl cunoaştem, putem folosi acest nume ca mai jos:

```
Debug.Print CurrentDB.TableDefs("Profesor").RecordCount
```

Am văzut că pentru a obţine baza de date curentă, trebuie să scriem:
`DBEngine.Workspaces(0).Databases(0)` sau, pe scurt, `DBEngine(0)(0)`

Access mai pune la dispoziţie şi funcţia CurrentDB(), pentru a obţine un pointer la baza de date curentă. Dacă dorim să accesăm o bază de date din afara sistemului Access (prin OLE), trebuie să folosiţi varianta DBEngine(0) (0). Altfel, dacă lucrăm în Access, putem folosi ambele variante, cu observaţia că funcţia CurrentDB() este mai rapidă.

Atenţie, însă, la următorul aspect: nu putem folosi un pointer la un obiect pe care l-aţi obţinut apelând la funcţia CurrentDB() într-o linie următoare de cod. Cu alte cuvinte, următoarea procedură:

```
Sub Test1()
Dim doc As Document
Set Doc = CurrentDb.Containers!Forms.Documents(0)
Debug.Print doc.Name
End Sub
```

va eşua când va încerca să tipărească doc.Name deoarece, în acel punct, doc este un pointer invalid. Următoarea procedură, în schimb, va rula corect:

```
Sub Test2()
Dim db As Database
Dim doc As Document
Set db = CurrentDb()
Set Doc = db.Containers!Forms.Documents(0)
Debug.Print doc.Name
End Sub
```

Folosirea semnului exclamării (!) şi a punctul (.)

Atât operatorul (!), cât şi operatorul (.) sunt folosiţi pentru a descrie relaţiile de apartenenţă dintre colecţii, obiecte şi proprietăţi.
In general, după (!) urmează numele unui obiect creat de utilizator un formular, un raport sau un control (deci un element al unei colecţii). Astfel, operatorul (!) separă un obiect din colecţia din care face parte. Operatorul (.) este folosit, în general, pentru a separa un obiect de o colecţie, proprietate sau metodă a sa.

3. Colecţii implicite

Am putut observa din exemplele de până acum că adresa, pentru a obţine un pointer la un obiect, trebuie să scrieţi o linie de cod destul de lungă. In Access însă, aproape orice tip de obiect are o colecţie implicită, care va fi luată în considerare dacă nu specificaţi nici o colecţie. De aceea,

`DBEngine.Workspaces(0).Databases(0).TableDefs(0)` se mai poate scrie prescurtat ca: `DBEngine(0)(0)`

Aceasta pentru că, după cum putem vedea şi în tabelul următor, TableDefs e colecţia implicită a unui obiect de tip Databases e colecţia implicită a unui obiect de tip Workspace, iar Workspaces e colecţia implicită a obiectului DBEngine.

Obiect	Colecţia implicită
Container	Documents
Database	TableDef
DBEngine	Workspaces
Group	User
Index	Fields
QueryDef	Parameters
Recordset	Fields
Relation	Fields
TableDef	Fields
User	Groups
Workspace	Databases

4. Lucrul cu proprietăţile obiectelor

Fiecare obiect DAO are o colecţie de proprietăţi. Unele obiecte au şi proprietăţi care nu există până în momentul în care li se dă o valoare. De aceea, este important să înţelegeţi diferenţele dintre tipurile de proprietăţi ale obiectelor DAO.

Tipuri de proprietăţi

Există două tipuri de proprietăţi DAO: *predefinite* şi *definite de utilizator*.
Proprietăţile predefinite ale unui obiect reprezintă caracteristicile de bază ale acestuia. Ele există încă de la crearea obiectului şi sunt disponibile pentru orice aplicaţie ce foloseşte motorul Jet. De exemplu, pentru un obiect de tip Field (camp), proprietăţile Name (nume), şi Type (tip) sunt predefinite.
Proprietăţile definite de utilizator nu există până când nu sunt adăugate la colecţia Properties a obiectului. De exemplu, proprietatea Description (descriere) a unui câmp nu este predefinită şi nu există până când nu introduceţi descrierea. Dacă încercaţi să regăsiţi proprietatea Description a unui obiect ce încă nu o are, veţi obţine o eroare.

Accesul la proprietăţi

Pentru a regăsi o proprietate predefinită a unui obiect, putem folosi sintaxa: *obiect.proprietate*
Pe de altă parte, nu putem regăsi o proprietate definită de utilizator decât prin intermediul colecţiei Properties a obiectului:
obiect.Properties("proprietate") sau *obiect.Properties!proprietate*
Această din urmă sintaxă funcţionează şi pentru proprietăţi predefinite.

Stabilirea proprietăţilor

Cu ajutorul instrucţiunii *With* putem stabili mai multe proprietăţi ale unui obiect, fără a mai fi necesar să specificăm obeictul pentru fiecare proprietate în parte:

```
Private Sub Set_Form_Prop()
With txtCtl
       .Text = "Acesta este un control de tip text"
       .ForeColor = RGB(0,  255,  0)      'verde
       .BackColor = 0
End With
End Sub
```

Crearea şi manipularea obiectelor cu DAO

In exemplele de până acum am folosit DAO pentru a accesa obiecte existente şi proprietăţile acestora. O mare parte a facilităţilor oferite de DAO constă însă în posibilitatea de a crea şi de a modifica obiecte.

Crearea obiectelor

Pentru a crea un obiect nou, procedăm astfel:

1. Folosim una dintre metodele Create… (CreateTable, CreateIndex etc.) pentru a crea obiectul dorit (tabelă, index etc.).
2. Stabilim proprietatea obiectului creat. Unele proprietăţi (cum ar fi numele) sunt esenţiale şi trebuie specificate la crearea obiectivului. Altele pot fi stabilite şi ulterior.
3. Adăugăm obiectul la colecţia corespunzătoare, pentru ca el să facă parte din baza de date.

Dacă obiectul nou creat conţine alte obiecte (aşa cum o tabelă conţine câmpuri), trebuie să creăm mai întâi obiectul principal şi pe urmă pe cele subordonate, pe care le adăugăm la colecţiile corespunzătoare ale obiectului principal. Apoi, îl adăugăm şi pe acesta la colecţia sa. Tabelul următor prezintă în detaliu metodele *Create*… cu argumentele acestora şi descrierile.

Obiect	Metoda	Argumente	Tip de date	Descriere
Tabela	CreateTableDef	Name	String	Numele tabelei
		Attributes	Integer	Stabilire valori pentru tabele atasîate, sistem sau ascunse.
		Source	String	Informaţii despre tipul tabelei de bază a unei tabele ataşate
		Connect	String	Calea şi numele fişierului unei tabele ataşate
Câmp	CreateField	Name	String	Numele câmpului
		Type	Integer	Tipul de date
		Size	Integer	Dimensiunea (pentru câmpuri de text text)
Index	CreateIndex	Nume	String	Numele indexului
Interogare	CreateQueryDef	Name	String	Numele interogării
		SQL	String	Sir de caractere ce conţine instrucţiunea SQL pe care se bazează interogarea
Relaţie	CreateRelation	Name	String	Numele relaţiei
		Table	String	Numele tabelei primare
		ForeignTable	String	Numele tabelei ce conţine cheia străină
		Attributes	Integer	Atributele relaţiei: tipul relaţiei,integritatea referenţială, actualizări şi ştergeri în casacadă
Workspace	Createspace	Name	String	Numele sesiunii de lucru
		User	String	Numele utilizatorului
		Password	String	Parola pentru sesiune
Bază de date	CreateDatabase	DatabaseName	String	Numele fişierului ce conţine baza de date
		Locale	String	Expresie ce specifică limba pentru noua bază de date (pentru sortări şi comparaţii între variabile de tip text)
		Options	Integer	O constantă sau o combinaţie de constante ce arată dacă baza de date este criptată sau versiunea motorului Jet
Grup	CreateGroup	Name	String	Numele grupului de utilizatori
		PID	String	Identificatorul grupului
Utilizator	CreateUser	Name	String	Numele utilizatorului
		PID	String	Identificatorul utilizatorului

Crearea unei tabele

Funcţia următoare creează tabela Profesori2 şi îi adaugă doar două coloane: IdProf şi Nume.

```
Function CreeazaTabProf()
Dim db As Database
Dim tbProf As TableDef
Dim fd1 As Field
Dim fd2 As Field
Set db = CurrentDb()
'Crearea tabelei si a coloanelor
Set tdProf = db.CreateTableDef()
tdProf.Name = "Profesori2"
Set fd1 = tdProf.CreateField ("IdProf", dbLong)
Set fd2 = tdProf.CreateField ("Name", dbText, 60)
'Adăugarea la colecţii
With tdProf.Fields
       .Append fd1
       .Append fd2
End With
With db.TableDefs
       .Append tdProf
       'Actualizarea colecţiei TableDefs
       .Refresh
End With
End Function
```

Este bine să folosim metoda Refresh (actualizare) pentru a vă asigura că noile obiecte au fost adăugate (acest lucru e recomandat mai ales în cazul unei aplicaţii multiuser).

Notă: Această funcţie nu tratează nici o eroare şi, de aceea, va eşua dacă o vom rula de două ori consecutiv. Pentru a o putea rula cu succes a doua oară, ştergem mai întâi tabela creată.

Crearea unui index

Paşii care trebuie urmaţi pentru a crea un index folosind DAO:

1. Folosim metoda CreateIndex a unui obiect de tip TableDef pentru a crea indexul şi stabiliţi-i proprietatea Name (nume);
2. Atribuim valorile dorite proprietăţilor noului index (cele mai importante proprietăţi sunt: Name,Primary, Unique şi Required). După ce am adăugat indexul la colecţia sa, nu-i vom mai putea modifica proprietăţile. Pentru aceasta, va trebui să ştergem obiectul şi să creem unul nou, cu proprietăţile dorite.
3. Folosim metoda CreateField a indexului pentru a crea câte un obiect de tip Field pentru fiecare câmp ce face parte din index. Adăugaţi-le apoi la colecţia Fields a indexului.
4. Folosim metoda Append a obiectului de tip TableDef pentru a adăuga indexul la colecţia Indexes.

Exemplu: Funcţia *CreareCheiePrimara()* creează o cheie primară pentru o tabelă. Ea primeşte ca argumente numele tabelei, numele indexului şi coloanelor ce vor compune cheia primară. Argumentul varColoane fiind de tip Variant, poate conţine fie o matrice, dacă vor fi mai multe coloane în componenţa cheii primare, fie un singur nume. Pentru a testa dacă nu există deja o cheie primară pentru tabela respectivă, funcţia CreareCheiePrimara() apelează funcţia ExistaCP(), care returnează numele indexului corespunzător cheii primare, dacă acesta există, sau valoarea Null, dacă nu există.

```
Function CreareCheiePrimara(strTabela As String, strCP As String,_ varColoane As Variant) As Boolean
Dim db As Database
Dim td As TableDef
Dim idxs As Indexes
Dim idx As Index
Dim fd As Field
Dim varCp As Variant
Dim varIdx As Variant
On Error GoTo CreareCheiePrimara_Err
Set db = CurrentDb()
Set td = db.TableDefs(strTabela)
Set idx = td.Indexes
'dacă există o cheie primară, o ştergem
varCP = ExistaCP(td)
If Not IsNull(varCP) Then
       Idxs.Delete varCp
End If
„cream noul index
Set idx = td.CreateIndex(strCP)
'facem ca indexul să fie cheie primara. Astfel, stabilim automat proprietatile:
```

```
'ignoreNulls = False; Required = True; Unique = True
idex.Primary = True
'cream campurile si le adaugam la colectia Fields
'dacă avem mai multe coloane:
If IsArray(varColoane) Then
   For Each varIdx In varColoane
       Call AdaugareCamp(idx, varIdx)
   Next varIdx
Else
'daca avem o singura coloana:
       Call AdaugareCamp(idx, varColoane)
End If
'adaugam indexul la colectia Indexes
Idxs.Addend idx
CreareCheiePrimara = True
CreareCheiePrimara_Exit:
   Exit Function
CreareCheiePrimara_Err:
   MsgBox "Crearea Indexului a esuat"
   CreareCheiePrimara = False
Resume CreareCheiePrimara_Exit
End Function

Private Function ExistaCP(td As TableDef) As Variant
Dim idx As Index
For Each idx In td.Indexes
   If idx.Primary Then
       ExistaCP = idx.Name
       Exist Function
   End If
Next idx
ExistaCP = Null
End Function

Private Function AdaugareCamp(idx As Index, varIdx As Variant)_
   As Boolean
Dim fd As Field
On Error GoTo AdaugareCamp_err
If Len(varIdx & " ") > 0 Then
   Set fd = idx.CreateField(varIdx)
   idx.Fields.Append fd
End If
AdaugareCamp = True

AdaugareCamp_Exit:
   Exit Function
AdaugareCamp_Err:
   AdaugareCamp = False
   Resume AdaugareCamp_Exit
End Function

Sub TestCP()
Debug.Print CreareCheiePrimara("Curs_Prof", "PrimaryKey",Array("IdCurs", "IdProf"))
End Sub
```

Crearea unei relaţii

Pentru a crea o relaţie folosind DAO, trebuie să efectuăm următorii paşi:

1. Verificăm dacă tabela primară (tabela din partea 1 a relaţiei) are definită o cheie primară.
2. Folosim metoda CreateRelation a obiectului bază de date pentru a crea relaţia. Proprietăţile relaţiei le puteţi stabili fie la crearea ei, fie ulterior, una câte una (Table, ForeignTable şi Attribute sunt printre cele mai importante).
3. Creăm câte un obiect de tip Field (câmp) pentru fiecare coloană din cheia primară a tabelei primare. Pentru fiecare dintre acestea, stabiliţi proprietatea ForeignName, ce reprezintă numele coloanei corespunzătoare din tabela din partea (many) a relaţiei. Adăugaţi apoi obiectele de tip Field la colecţia Field a relaţiei.
4. Folosim metoda Append a obiectului bază de date pentru a adăuga relaţia la colecţia Relations.

Exemplu: Funcţia CreareRel, prezentată mai jos, creează o asociere left outer join tabelele Profesor şi Titlu şi permite actualizările în cascadă. Dacă relaţia există deja, funcţia CreareRel o şterge şi o creează din nou.

```
Function CreareRel() As Boolean
Dim db As Database
Dim rel As Relation
```

```
Dim fd As Field
On Error GoTo CreareRel_Err
Set db = CurrentDb()
'cream obiectul relatie
Set rel = db.Createrelation()
'stabilim proprietatile relatiei
With rel
       .Name = "RelProfTitlu"
       .Table = "Titlu"
       .ForeignTable = "Profesor"
   'left outer join, modificari in cascada
   .Attributes = dbRelationLeft Or dbRelationDeleteCascade
  End With
   'cream campurile
Set fd = rel.CreateField("IdTitlu")
Fd.ForeignName = "IdTitlu"
Rel.Fields.Append fd
MsgBox "S-a creat relatia"
'adaugam relatia la colectia Relations
Db.Relations.Append rel
CreaRel = True
CreareRel _Exit:
   Exit Function
CreareRel_Err:
    Select Case Err.Number
       Case 3012 'daca relatia exista deja
            Db.Relations.Delete rel.Name
             Resume
       Case Else
             MsgBox "Relatia nu a putut fi creata"
             CreareRel = False
             Resume CreareRel_Exit
    End Select
End Function
```

Crearea proprietăţilor-utilizator

DAO vă dă posibilitatea să creaţi noi proprietăţi pentru un obiect, pe care să le adăugaţi apoi la colecţia Properties a obiectului respectiv. Paşii care trebuie urmaţi sunt cei de mai jos:

1. Folosim metoda CreateProperty a obiectului respective pentru a crea proprietatea.
2. Definim caracteristicile proprietăţii.
3. Adăugăm proprietatea la colecţia Properties a obiectului.

Exemplu: Se creează o nouă proprietate pentru o tabelă: UltimaModificare. Ea păstrează data ultimei modificări a tabelei.

```
Sub UltimaModifi(strTab As String)
Dim db As Database
Dim td As TableDef
Dim prUltMod As Property
Set  db = CurrentDb()
Set td = db.TableDefs(strTab)
'cream noua proprietate
Set prUltMod = td.CreateProperty("UltimaModificare",dbDate,New)
'adăugăm UltimaModificare la colecţia Properties
'ignorăm posibilele erori
On Error Resume Next
Td.Properties.Append prUltMod
'afişăm valoarea proprietăţii
Debug.Print.tdProperties("UltimaModificare"). Value
End Sub
```

Modificarea obiectelor

Puteţi modifica un obiect existent fără a-l deschide în modul Design View, ci numai folosind proprietăţile şi metodele furnizate de DAO. Trebuie totuşi să ţineţi cont de anumite restricţii atunci când stabiliţi proprietăţile unui obiect DAO. Unele proprietăţi pot fi stabilite numai la crearea obiectului şi nu mai pot fi modificate după ce acesta a fost adăugat la colecţia corespunzătoare (proprietatea Attributes a unui obiect de tip TableDef e un exemplu în acest sens). Pentru a modifica o astfel de proprietate, trebuie să creaţi un nou obiect, să copiaţi în el toată informaţia conţinută în cel vechi, să stabiliţi proprietatea respectivă, să adăugaţi noul obiect la coleţia sa şi apoi să ştergeţi vechiul obiect. De asemenea, trebuie să ţineţi cont de faptul că unele proprietăţi ale unui obiect DAO nu există decât din momentul în care li s-a dat o valoare, astfel încât încercarea de a regăsi o astfel de proprietate s-ar putea solda cu o eroare.