# Metode avansate de acces la date utilizând ADO

Lect.dr. Cornelia MUNTEAN
Facultatea de Stiinte Economice, Universitatea de Vest din Timisoara

*This paper presents Microsoft's ActiveX Data Objects model, a model of choice for future database development within Visual Basic. ADO is a fast, high-level interface to the OLE DB application – programming interface (API). The most used ADO control is the ADO Data control, used for creating very rapidly connections between the data bound controls and the data sources. Finally a database application using the ADO Data control is presented, with some code examples for handling ADO.*
***Keywords**: data objects, interface, data base, application.*

## Tehnologia ADO

Modelul cel mai nou al lui Microsoft de accesare a bazelor de date utilizeaza obiecte de date ActiveX (ADO – **A**ctiveX **D**ata **O**bjects) si promite a fi metoda de viitor pentru dezvoltarea bazelor de date cu Visual Basic. Acest model s-a nascut din dorinta de a avea un acces universal la date (UDA – Universal Data Access), asa cum numeste Microsoft ideea ca un dezvoltator sa fie capabil de a folosi o metoda de acces la date indiferent de sursa de date pe care o interogheaza. ODBC (**O**pen **D**atabase **C**onnectivity), care constituie varianta Microsoft a limbajului SQL pentru mediul client/server a reprezentat un mare pas înainte. Pentru prima data, indiferent de baza de date cu care aplicatia trebuie sa „converseze", era necesar ca utilizatorul sa cunoasca o singura interfata API (**A**pplication **P**rogram **I**nterface). Marele dezavantaj era însa faptul ca ODBC era orientat direct spre bazele de date relationale, iar celelalte surse de date nu se încadrau deloc bine în modelul sau. Astfel a luat nastere OLE DB, o interfata de tip COM (**C**omponent **O**bject **M**odel) între furnizorii de date (aflati în general pe servere la distanta) si aplicatiile client. În OLE DB furnizorii de date pot fi orice obiecte, de la baze de date relationale la foi de calcul tabelar sau sisteme de fisiere. Este important totusi a sublinia faptul ca OLE DB nu înlocuieste ODBC, deoarece ele prezinta interfete API complet independente. Oricum, în viitorul foarte apropiat se asteapta ca OLE DB sa depaseasca ODBC în ceea ce priveste usurinta de utilizare si functionalitatea.

În concluzie, ADO este interfata din OLE DB pentru ca Visual Basic sa poata beneficia de avantajele accesului universal la date (UDA), deci sa nu fie restrictionat la surse de date Jet, ISAM, sau chiar baze de date relationale. ADO este rapida si sintaxa sa se aseamana foarte mult cu mai vechiul model DAO (**D**ata **A**ccess **O**bjects), astfel încât programatorii DAO vor putea face trecerea usor spre noua tehnologie de acces la date cu ADO.

Principalele avantaje pe care le ofera aplicatiile elaborate utilizând ADO sunt: viteza mare; memorie redusa; usurinta în utilizare; facilitati pentru construirea aplicatiilor client/server si cele web.

În ADO nu mai este necesara navigarea printr-o ierarhie de obiecte pentru a crea un obiect derivat din obiectul radacina al clasei. Atunci când în DAO, de exemplu, trebuia creat un set de înregistrari, aceasta însemna ca mai întâi sa se deschida o conexiune la baza de date prin intermediul unui obiect baza de date si sa se creeze un obiect Recordset din obiectul baza de date. Practic trebuia sa dispunem de toate obiectele parinte. ADO nu necesita suprasarcina unei ierarhii. Pentru a regasi rezultate dintr-o sursa de date, nu este nevoie decât de o conexiune si de un set de înregistrari.

Modelul de obiecte de date ActiveX contine o ierarhie de sapte obiecte (figura 1).

Cu toate ca obiectele ADO pot fi create în afara domeniului unei ierarhii, obiectele exista în cadrul relatiilor ierarhice. Oricând poate fi creat un obiect independent, de exemplu un

set de înregistrari cu sau fara obiectul Connection. Acest fapt reduce obiectele care trebuie gestionate prin program la:

- *Obiectul Connection*, care gestioneaza o conexiune cu un furnizor de date ADO. Obiectul Connection este radacina modelului ADO. Din acesta pot fi derivate toate celelalte sase obiecte.

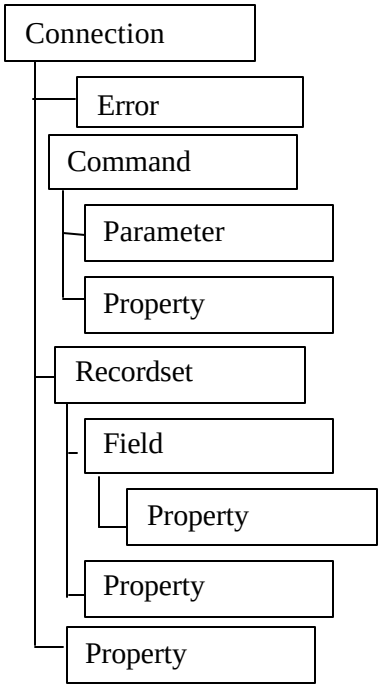



**Fig. 1.** Modelul obiect ADO

- *Obiectul Command*, care contine proprietatile si metodele necesare pentru executia comenzilor SQL folosind ADO.
- *Obiectul Error*, care apartine colectiei Errors, gestioneaza erorile generate de furnizorul de date pentru un obiect Connection.
- *Obiectul Parameter*, apartine colectiei Parameters si este folosit cu obiectul Command pentru a executa proceduri stocate.
- *Obiectul Recordset*, care trateaza rândurile rezultate dintr-un tabel deschis sau dintr-o interogare executata.
- *Obiectul Field*, apartinând colectiei Fields, reprezinta coloanele sau câmpurile obtinute într-un obiect Recordset.
- *Obiectul Property*, apartinând colectiei Properties, gestioneaza caracteristicile dinamice ale unui obiect ADO.

Scopul principal al filozofiei COM este reutilizarea obiectelor odata create în mai multe aplicatii. Urmând principiile COM, aplicatiile sunt construite din mai multe componente obiect care sunt legate împreuna cu un limbaj de programare (ca Visual Basic).

**Folosirea controlului ADO Data**

Controlul ADO Data utilizeaza Microsoft ADO pentru a crea rapid legaturi între controalele asociate datelor si sursele de date. Controalele asociate datelor sunt oricare care poseda proprietatea DataSource. Sursele de date sunt surse realizate conform specificatiilor OLE DB. Controlul ADO Data are acelasi aspect ca si controlul Data incorporate (figura 2) – un set de patru butoane prin care utilizatorul poate ajunge imediat la începutul setului de înregistrari, la sfârsitul sau si poate naviga înainte si înapoi în cadrul setului.


**Fig. 2.** Controlul Data si controlul ADO Data

Datorita flexibilitatii din ADO se recomanda ca noile aplicatii de tip baza de date sa fie realizate cu ajutorul controlului ADO Data în locul controlului încorporat Data sau a controlului **R**emote **D**ata (**RDC**), disponibile ambele din mai vechile versiuni de Visual Basic.

Posibilele utilizari ale controlului ADO Data sunt:

- conectarea la o baza de date locala sau la distanta;
- deschiderea unei tabele dintr-o baza de date, definirea unui set de înregistrari bazat pe o interogare de tip SQL, o procedura stocata sau vizualizarea tabelelor din baza de date;
- transferarea de valori ale câmpurilor de date înspre controale asociate datelor, acolo unde ele pot fi afisate sau li se poate modifica valoarea;
- adaugarea de noi înregistrari sau actualizarea unei baze de date, bazata pe schimbarile facute la nivelul datelor afisate în controalele asociate.

Pentru a crea o baza de date client sau pentru un utilizator final se adauga controlul ADO Data la formular, împreuna cu controalele incorporate necesare care pot fi asociate con-

trolului ADO Data. Practic orice control care poseda proprietatea DataSource poate fi asociat unui control ADO Data.

**Prezentarea aplicatie i**

S-a realizat o aplicatie didactica în Visual Basic de calcul salarial având un proiect cu zece forme, un modul de cod si trei rapoarte, realizate cu Microsoft Data Report (figura 3).

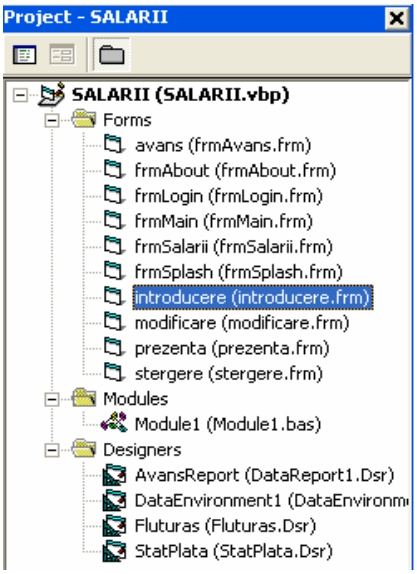



**Fig. 3. A**plicatia de calcul al salariilor

Formele *avans*, *salarii*, *introducere*, *modificare* si *stergere* contin fiecare câte un control ADO Data. Acesta e conectat prin intermediul furnizorului de date Microsoft Jet 3.51 OLE DB Provider la baza de date DATE.MDB, a carei cale completa trebuie data la proprietatea Connectionstring a controlului ADO Data (figura 4).

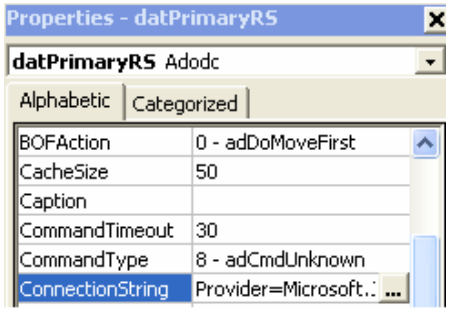



**Fig. 4.** Setarea proprietatii ConnectionString

Selectând butonul *Build...* apare fereastra *Data Link Properties* (figura 6), în care din pagina Connection se alege baza de date .MDB dorita (în cazul prezentat D:\...\DATE.MDB) si numele utilizatorului (de exemplu "Admin"), iar în pagina Provider s-a ales Microsoft Jet 3.51 OLE DB Provider.

La clic pe butonul de completare a proprietatii asociata lui *ConnectionString* apare pe ecran fereastra *Property Pages* (figura 5).

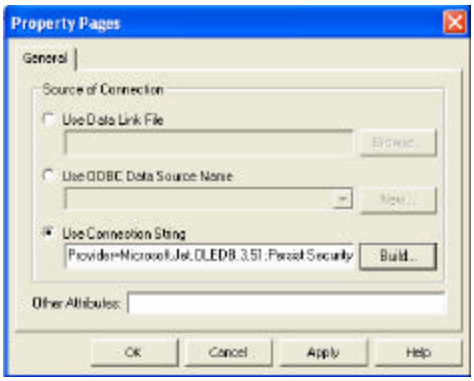



**Fig. 5.** Fereastra Property Pages

La clic pe butonul *Test Connection*, daca toate setarile de conexiune s-au facut corect, va apare o fereastra de mesaj cu textul: "Test connection succeeded".

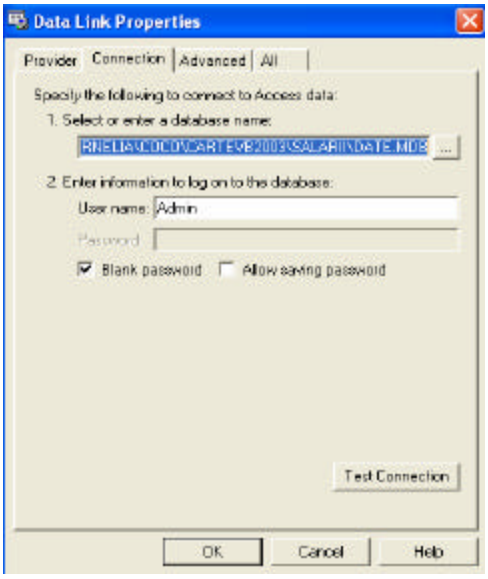
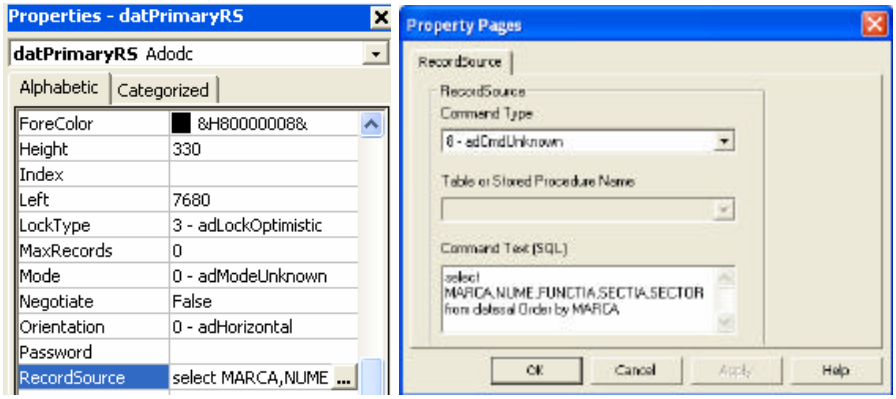



**Fig. 6.** Fereastra Data Link Properties

În cea ce priveste proprietatea *Recordsource*, în multe din formele aplicatiei se va selecta un set de înregistrari cu o comanda de interogare SQL SELECT (figura 7 a).




**a)** **b)**

**Fig. 7.** Setarea proprietatii RecordSource

Comanda se scrie fie direct în spatiul prevazut pentru aceasta, fie se deschide fereastra de dialog asociata proprietatii (figura 7 b) si se scrie comanda în caseta *Command Text (SQL)* dând apoi clic pe butonul *OK*.

Forma prezenta, de exemplu, nu contine controlul ADO Data (figura 8), ci doar un control DataGrid, care însa este legat prin linii de cod la sursa de date la încarcarea formei, respectiv la alegerea unuia din butoanele de optiune care filtreaza datele pentru ca în lista afisata în DataGrid sa apara doar persoane apartinând unei anumite sectii.

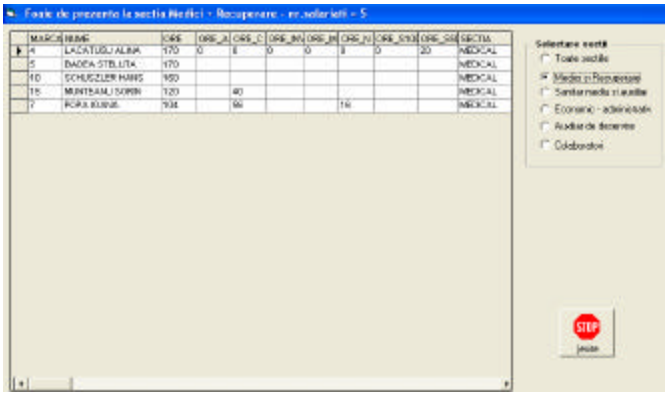


**Fig. 8.** Forma *prezenta*, continând controlul DataGrid

În acest caz procedura *Form_Load()*, respective *med_Click()* (corespunzatoare optiunii sectiei Medici si Recuperare) ar fi cele din liniile de mai jos:

```
Private Sub Form_Load()
 Dim db As Connection
 Set db = New Connection
 db.CursorLocation = adUseClient
 db.Open "PRO-
VIDER=Microsoft.Jet.OLEDB.3.51;Data
Source=D:\CORNELIA\COCO\carteVB2003\SALARII\D
ATE.MDB;"
 Set adoPrimaryRS = New Recordset
 adoPrimaryRS.Open "select
MARCA,NUME,ORE_A,ORE_C,ORE_INV,ORE_M,ORE_N,OR
E_S100,ORE_S50,SECTIA from datesal Order by
MARCA", db, adOpenStatic, adLockOptimistic
 Set grdDataGrid.DataSource = adoPrimaryRS
 mbDataChanged = False
End Sub
'*****************************************
************************
Private Sub med_Click()
ns = 0
med.Value = True
Dim db As Connection
 Set db = New Connection
 db.CursorLocation = adUseClient
 db.Open "PRO-
VIDER=Microsoft.Jet.OLEDB.3.51;Data
Source=D:\CORNELIA\COCO\CARTEVB2003\SALARII\D
ATE.MDB;"
Set adoPrimaryRS = New Recordset
adoPrimaryRS.Open "select
MARCA,NUME,ORE,ORE_A,ORE_C,ORE_INV,ORE_M,ORE_
N,ORE_S100,ORE_S50,SECTIA from datesal WHERE
sectia='MEDICAL' ", db, adOpenStatic, ad-
LockOptimistic
 Set grdDataGrid.DataSource = adoPrimaryRS
 mbDataChanged = False
ns = adoPrimaryRS.RecordCount
prezenta.Caption = " Foaie de prezenta la
```

```
sectia Medici + Recuperare - nr.salariati = "
& ns
prezenta.Refresh
End Sub
```

## Concluzii

Cu controlul ADO Data, care utilizeaza relativ noua tehnologie Microsoft ActiveX Data Objects se pot crea rapid controalele asociate datelor si sursele de date. Controlul ADO Data este indicat în aplicatii rapide si simple, însa pentru a beneficia de caracteristici avansate va trebui sa se treaca la manipularea prin program a ADO. Astfel se poate obtine acces la date fara a utiliza în mod direct un control ADO Data asociat.

## Bibliografie

1. E. Winemiller, J.T. Roff, B. Heyman, R. Groom, „Visual Basic 6. Baze de date”, Editura Teora, 2001
2. * * *, Microsoft Visual Basic 6.0. Ghidul programatorului, Editura Teora, 2003
3. Doug Lowe, Tehnologia client/server pentru toti, Editura Teora, 1996
4. P. Norton, „Peter Norton’s Guide to Visual Basic 6”, Macmillan Computer Publishing, 1998
5. C. Muntean, P.Stepanov, O.Dobrican, „Visual Basic. Aplicatii rezolvate pas cu pas”, Editura Mirton, 2002
6. „Teach Yourself Database programming with Visual Basic in 21 Days”, documentatie IT