Tehnologia ADO în mediul Borland Delphi

stud. Corneliu V. Bogdan, conf. dr. ing. Dan L. Lacrămă
Universitatea „Tibiscus" Timişoara

ABSTRACT. This article focused on the use of the ADO technology in Borland Delphy for connecting a wide range of data sources in Windows and Linux. The set of ADO interfaces is presented in order to draw a complete image of their utility in different applications.

1 Generalităţi

Încă de pe la jumătatea anilor '80, s-a căutat o soluţie pentru independentă faţă de bazele de date. Ideea era să se folosească un singur set API astfel încât aplicaţiile să poată interacţiona cu mai multe surse de date diferite. Astfel se evită dependenţa de un singur motor de baze de date. Cele mai notabile dintre primele soluţii au fost Microsoft Open Database Connectivity (ODBC) şi Borland Integrated Database Application Programming Interface (IDAPI) mai bine cunoscut sub numele Borland Database Engine (BDE).

Pe la mijlocul anilor '90, odată cu succesul modelului Component Object Model (COM), Microsoft a început să înlocuiască ODBC-ul cu Object Linking and Embeding Data Base (OLE DB). OLE DB este un set de interfeţe bazate pe COM care asigură aplicaţiilor acces uniform la datele de la diferite surse de informaţie. Aceste interfeţe suportă specificităţile diferitelor SGBD-uri, facilitând astfel accesul la înregistrările dorite. OLE DB sunt interfeţe de nivel sistem şi în consecinţă sunt folosite doar de programatorii de sistem, fiind foarte mari şi complexe. ActiveX Data Objects (ADO) este un strat "aşezat" peste OLE DB şi este referit ca interfaţă de nivel aplicaţie.

Microsoft ADO asigură acces aplicaţiilor client la date şi manipularea acestora prin intermediul unui provider OLE DB. Principalele

beneficii sunt uşurinţa în folosire, viteza mare, amprentă mică în memorie şi de spaţiul mic ocupat pe disc. De asemenea, ADO suportă funcţiile cheie necesare pentru construirea de aplicaţii client-server pe Internet şi intranet.

ADO cuprinde, de asemenea, funcţiile Remote Data Service (RDS), prin care se pot muta datele de pe server pe o aplicaţie client sau pagină WEB, manipularea acestora pe client şi întoarcerea modificărilor pe server printr-o singura "mişcare".

Interfaţa ADO este creată pentru a da programatorilor un puternic model obiectual cu care prin programare să se poată accesa şi modifica o mare varietate de surse de date prin intermediul interfeţelor OLE DB. Prin urmare prin intermediul ADO programatorul poate:

- vizualiza tabele din BD şi să facă modificări în acestea;
- interoga o baza de date şi afişa rezultatele;
- accesa informaţiile prin Internet;
- manipula mesajele şi directoarele într-un sistem de e-mail;
- salva datele dintr-o bază de date într-un fişier XML;
- execută comenzi descrise prin XML şi să primească unui flux XML;
- salva datele într-un flux binar sau XML;
- crea şi refolosi comenzi parametrizate pentru bazele de date;
- execută proceduri stocate;
- crea dinamic structuri flexibile pentru a manipula date;
- execută tranzacţii în bazele de date;
- execută filtrări şi sortări "run-time" ale informaţiilor;
- crea şi manipula rezultate ierarhice din bazele de date.

Cea mai utilizată funcţie a ADO este să facă interogări pe o tabelă sau mai multe dintr-o bază de date relaţională pentru a extrage şi afişa rezultatele într-o aplicaţie.

Datorită faptului că ADO este parte din Microsoft Data Access Components (MDAC), care conţine ADO, OLE DB, ODBC şi RDS apar unele avantaje. În primul rând este foarte probabil ca utilizatorii aplicaţiilor create să aibă deja instalat MDAC pe sisteme. Apoi este sigur că utilizatorii vor avea aproape tot timpul cea mai recentă versiune de MDAC instalată deoarece MDAC se instalează împreună cu componente foarte folosite de genul Internet Explorer.

Componentele facilitează conectarea la sursele de date suportate de ADO, execuţia de comenzi şi extragerea de date din tabelele bazelor de date. Adiţional, programe client trebuie instalate în funcţie de sistemul de baze de date folosit (cum ar fi Microsoft SQL Server), împreună cu drivere OLE DB sau ODBC specifice. Componentele existente sunt:

- ***TADOConnection:*** componenta pentru conectarea la baza de date. Mai multe datasets pot folosi în comun această conexiune pentru a executa comenzi, pentru a extrage şi opera date.
- ***TADODataSet:*** principala componentă pentru extragerea şi a operarea datelor. Datele se pot selecta din una sau mai multe tabele, iar conexiunea se poate face direct la sursa de date sau printr-o componentă TADOConnection.
- ***TADOTable:*** un dataset de tip tabel folosit pentru extragerea şi operarea unui set de înregistrări produs de o singură tabelă din baza de date. Poate fi conectat direct la o sursă de date sau se poate folosi de o conexiune existentă.
- ***TADOQuery:*** un dataset de tip interogare care poate extrage un set de înregistrări bazat pe o comandă validă SQL. Poate executa şi comenzi DDL (Data Definition Language) La fel ca şi celelalte componente poate fi folosit fie singur fie împreună cu o componentă TADOConnection.
- **TADOStoredProc:** un dataset pentru execuția procedurilor stocate care pot sau nu să genereze date.
- ***TADOCommand:*** execută comenzi SQL care nu generează date. Poate fi folosită cu o altă componentă ADO de tip dataset.

Aceste componente compun pachetul de componente numite dbGo.

2 Conectarea la o sursă de date

Unul sau mai multe ADO datasets şi componente de comanda pot folosi împreuna o singură conexiune la o sursă de date utilizând TADOConnection. Pentru a se realiza acest lucru se va folosi proprietatea Connection a componentelor respective cu ajutorul ferestrei Object Inspector la proiectare. La rulare, se asociază proprietății Connection referința conexiunii (ex. ADODataSet1.Connection := ADOConnection1;).

TADOConnection reprezintă un obiect. Înainte de a se putea folosi conexiunea trebuie identificată sursa de date la care se doreşte conectarea. În mod normal se foloseşte proprietatea ConnectionString care este un şir de caractere ce foloseşte ";" pentru a delimita parametrii conexiunii. Se foloseşte următoarea configurație de parametrii:

- Provider: numele unui ADO provider utilizat cu conexiunea respectiva;
- Data Source: numele sursei de date;
- File name: numele unui fişier conținând informațiile despre conexiune;
- Remote Provider: numele unui ADO provider care există pe o altă stație;

- Remote Server: nume server distant în cazul în care se foloseşte o altă staţie.

Un exemplu de ConnectionString este:

```
Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\Program Files\Common Files\BorlandShared\Data\dbdemos.mdb;
Persist Security Info=False
```

Folosirea proprietăţii ConnectionString sub această formă poate să ridice anumite dificultăţi în cazul în care se foloseşte o sursă de date care defineşte bazele de date ca fişiere individuale pe disc, caz în care apar unele dificultăţi pentru a coda calea către acestea în executabil. De aceea se poate folosi un fişier Data Link care este de tip INI cu extensia UDL. De exemplu:

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;
  DataSource=C:\ProgramFiles\Common Files\Borland Shared\Data\dbdemos.mdb
```

Un asemenea fişier se poate crea foarte uşor folosind Notepad. Dublul click pe acest fişier în Windows Explorer va deschide Microsoft Connection String Editor. În proprietatea ConnectionString de la TADOConnection se scrie "FILE NAME = " urmat de numele fişierului UDL. Dacă se doreşte folosirea locaţiei standard a acestor fişiere în Windows se poate utiliza funcţia DataLinkDir din unit-ul ADODB care în mod implicit va returna calea "C:\Program Files\Common Files\System\OLE DB\Data Links".

„Schema information" se referă la felul şi forma în care este organizată informaţia în sursa respectivă de date. De exemplu, cataloage, seturi de caractere, coloane, restricţii, indecşi, privilegii, etc. În Delphi aceste date se pot extrage cu ajutorul metodei OpenSchema a componentei TADOConnection.

Pentru acest scop se poate folosi ADOX, care este o tehnologie adiţională pentru extragerea şi modificarea schemei sursei respective de date. Acesta este echivalentul lui ADO la DDL şi DML din SQL. ADOX nu este direct suportat de dbGO, dar prin folosirea ADOX type library se poate utiliza cu succes în aplicaţiile Delphi. Este totuşi recomandat ca pentru cazurile în care este necesară doar vizualizarea de informaţii fără modificare să se folosească metoda OpenSchema de la componenta TADOConnection deoarece ADOX nu este încă implementat pe scară largă.

Prin folosirea componentei TADOConnection se obţine un control mare asupra condiţiilor şi atributelor conexiunii. Proprietatea ConnectOptions se foloseşte pentru a forţa conexiunea să se comporte asincron. Conexiunile asincrone permit aplicaţiei să continue procesarea fără să aştepte deschiderea completă a conexiunii. Standard, această proprietate este setată cu valoarea coConnectUnspecified care lasă serverul

să decidă care este cel mai bun tip de conexiune. Pentru a face conexiunea asincronă se setează valoarea coAsyncConnect. Se poate controla şi perioada de timp care se poate scurge până când comenzile şi conexiunile sunt declarate ca eşuate şi pot fi anulate folosind proprietăţile ConnectionTimeout şi CommandTimeout.

Conexiunile ADO sunt stabilite folosind anumite moduri, similar cu cele în care se fac operaţiile cu fişiere. Aceste moduri determină permisiunile asupra conexiunii respective şi tipul operaţiilor care se pot face folosind acea conexiune. Pentru setare se foloseşte proprietatea Mode care poate avea următoarele valori:

- cmUnknown: permisiunile nu sunt setate sau nu pot fi determinate;
- cmRead: permisiuni Read-Only;
- cmWrite: permisiuni Write-Only;
- cmReadWrite: permisiuni de citire şi scriere;
- cmShareDenyRead: nu lasă mai multe conexiuni cu permisiuni de citire;
- cmShareDenyWrite: nu lasă mai multe conexiuni cu permisiuni de scriere;
- cmShareExclusive: conexiunea nu poate fi folosită de alţii;
- cmShareDenyNone: nu poate fi folosită de alţii cu orice permisiune.

Valorile acestei proprietăţi corespund valorilor ConnectModeEnum ale proprietăţii Mode din obiectul ADOConnection. Detaliile se găsesc în Microsoft Data Access SDK.

3 Folosirea ADO datasets

Componentele ADO dataset încapsulează obiectul ADO recordset. Ele moştenesc capabilităţile comune ale dataset-urilor din Delphi. Delphi foloseşte ca unitate fundamentală pentru accesarea bazelor de date familia de obiecte dataset. Un obiect dataset reprezintă un set de înregistrări dintr-o bază de date organizate într-o tabelă logică. Aceste înregistrări pot fi dintr-o singură tabelă a bazei de date sau pot reprezenta rezultatele executării unei interogări sau proceduri complexe.

Toate obiectele dataset din aplicaţie sunt descendente din TDataSet şi moştenesc câmpurile, proprietăţile, event-urile şi metodele acestei clase. TDataSet este un dataset virtual, însemnând că multe din proprietăţi şi funcţii sunt virtuale sau abstracte. O metodă virtuală este o funcţie sau procedură unde implementarea poate şi este de obicei rescrisă în obiectul descendent. O metodă abstractă este o funcţie sau procedură care nu are o

implementare. Este doar descriptivă, arată ce parametrii foloseşte şi ce returnează. Ea trebuie implementată în toți descendenții, dar poate fi implementată diferit în fiecare din ei.

Pe lângă facilitățile moştenite din clasa TDataSet, componentele ADO au proprietăți, event-uri şi metode pentru: conectare la un ADO datastore, accesarea obiectului Recordset aferent, filtrarea înregistrărilor bazate pe bookmarks, extragerea de înregistrări asincron, executarea de batch updates şi folosirea fişierelor de pe disc pentru stocarea datelor. Există patru tipuri de ADO datasets:

- TADOTable: reprezintă toate liniile şi coloanele unei singure tabele din BD;
- TADOQuery: încapsulează o comanda SQL şi acordă accesul aplicației la rezultatele acesteia;
- TADOStoredProc: execută o procedură stocată pe server;
- TADODataSet: combină capabilitățile celorlalte trei tipuri.

Când un dataset TADOQuery returnează un set de date, primeşte şi un cursor, sau un pointer către prima înregistrare din acel set de date. Înregistrarea indicată de cursor este înregistrarea activă în acel moment. Există două proprietăți fundamentale ale ADO dataset. CursorLocation şi CursorType.

CursorLocation: Prin această proprietate se specifică cine controlează extragerea şi modificarea datelor. Se poate selecta clientul (clUseClient) sau server-ul (clUseServer). Alegerea afectează funcționalitatea, performanța şi scalabilitatea dataset-ului respectiv.

CursorType: Un cursor de tip client este administrat de ADO Cursor Engine, adică toate datele din setul de înregistrări sunt extrase de pe server şi puse pe client când dataset-ul este deschis. Astfel datele sunt în memorie şi modificările sunt administrate de ADO Cursor Engine. Unul din beneficii este că manipularea datelor după extragerea inițială este foarte rapidă. Cursorul de tip server este administrat de SGBD. Într-o arhitectură client-server bazată pe un server de date SQL Server sau Oracle, cursorul este administrat fizic pe server. În aplicațiile de gen desktop cum ar fi Access sau Paradox, această locație este doar o locație logica deoarece baza de date este pe acelaşi calculator. De obicei aceste cursoare se încarcă mai repede deoarece datele nu sunt transferate pe client când dataset-ul este deschis. Acest aspect le face optime pentru cazurile în care sunt seturi foarte mari de date, când clientul are memorie insuficientă pentru a încărca tot pachetul de date. Un alt aspect care trebuie luat în calcul este scalabilitatea. În cazul cursoarelor de pe server, acesta poate ajunge să fie foarte încărcat datorită tuturor cursoarelor gestionate, rezultând ca SGBD-ul devine un ”bottleneck”

şi aplicaţia este mai puţin scalabilă. În astfel de cazuri se folosesc cursoare pe client. Deschiderea iniţială a cursorului este de obicei mai grea deoarece toate datele necesare sunt transferate pe client, în acelaşi timp managementul datelor poate deveni mai greoi.

4 Procesarea unei tranzacţii

Tranzacţiile permit gruparea modificărilor asupra înregistrărilor dintr-o bază de date într-o unitate logică de lucru. Componenta ADOConnection asigură managementul tranzacţiilor folosind metodele BeginTrans, CommitTrans şi RollbackTrans. Suportul pentru tranzacţii variază în funcţie de provider-ul OLE DB folosit. De exemplu, în cazul conectării la Paradox prin ODBC OLE DB va rezulta o eroare la folosirea acestor metode deoarece acest OLE DB provider nu suportă tranzacţii. în cazul în care se foloseşte Jet 4.0 OLE DB provider nu se va putea folosi funcţia rollback datorită limitărilor provider-ului. De asemenea, conectarea la Access prin ODBC OLE DB provider permite doar o tranzacţie odată. Deschiderea unei alte tranzacţii în timp ce o tranzacţie este activă generează o eroare. Această problemă se poate rezolva prin folosirea motorului Jet.

ADO datasets se conectează la un ADO datastore fie individual fie colectiv. În cazul conectării individuale se setează proprietatea ConnectionString a fiecărui dataset, astfel fiecare se conectează individual faţă de ceilalţi. Când se conectează colectiv se setează proprietatea Connection a fiecărui dataset cu valoarea unei componente TADOConnection.

ADODataset1.Connection:=ADOConnection1;

Avantajele conectării colective sunt: toate componentele dataset folosesc atributele obiectului conexiune, o singură conexiune trebuie setată, iar componentele pot participa la tranzacţii.

Proprietatea Recordset asigură acces direct la setul de înregistrări aferent componentei dataset. Prin folosirea acestui obiect se pot accesa proprietăţile şi executa metodele obiectului recordset din aplicaţie. Nu este recomandată folosirea directă a obiectului recordset doar în cazul în care utilizatorul este familiarizat cu operaţiile acestuia. Proprietatea RecordsetState indică starea curentă a recordset-ului aferent, valoarea acestuia corespunde proprietăţii State a obiectului ADO Recordset. Valoarea poate să fie stOpen, stExecuting sau stFetching. stOpen arată că recordset-ul este în aşteptare. stExecuting indică execuţia unei comenzi. stFetching arată că se extrag înregistrări din tabela sau tabelele asociate. Această proprietate

se foloseşte când se execută acţiuni dependente de starea curentă a dataset-ului.

Dataset-urile ADO suportă funcţia de a returna înregistrări care sunt marcate folosind bookmarks şi folosirea de filtre pentru a limita numărul de înregistrări extrase. De asemenea se poate face combinarea acestor doua metode, adică filtrarea unui set de înregistrări marcate. Pentru a realiza un asemenea filtru se foloseşte metoda Bookmark pentru a marca înregistrările care vor fi filtrate, apoi se execută metoda FilterOnBookmarks pentru a face filtrarea propriu-zisă.

Extragerea de înregistrări asincron permite aplicaţiei să execute alte task-uri în timp ce dataset-ul este populat cu date. Pentru controlul acestei facilităţi se foloseşte proprietatea ExecuteOptions care coordonează felul în care datele sunt returnate când se foloseşte metoda Open sau se setează proprietatea Active pe True. ExecuteOptions setează modul în care se execută comanda când se foloseşte metoda ExecSQL sau ExecProc. Proprietatea poate să nu ia nici o valoare sau una sau mai multe din următoarele :

- eoAsyncExecute: comanda este executată asincron
- eoAsyncFetch: extrage sicron numărul de înregistrări specificate de proprietatea CacheSize, apoi extrage alte înregistrări asincron
- eoAsyncFetchNonBlocking: execută comanda fără să blocheze thread-ul curent
- eoExecuteNoRecords: specifică o comandă care nu returnează date, dacă sunt generate înregistrări acestea nu sunt returnate.

Batch updates este un procedeu în care orice modificări ce se aduce înregistrărilor se face în memorie. Ulterior tot lotul de modificări poate fi executat ca o singură operaţie. Există avantaje practice şi de performanţă pentru această metodă. User-ul respectiv ar putea să nu fie conectat la baza de date în momentul în care fac modificările. Acest lucru se întâmplă în cazul aplicaţiilor de tipul briefcase sau în aplicaţiile web care folosesc tehnologia Remote Data Services (RDS).

Se poate folosi batch update în orice ADO dataset prin setarea proprietăţii LockType cu valoarea ltBatchOptimistic şi a proprietăţii CursorLocation cu valoarea clUseClient înainte ca dataset-ul să fie deschis. În acest fel se creează o listă de modificări, se fac modificările în memorie, iar dataset-ul "vede" înregistrările ca şi când acestea ar fi fost deja făcute. Pentru a face modificările permanente se foloseşte metoda UpdateBatch. Pentru a se anula modificările din memorie se foloseşte CancelBatch. UpdateStatus poate fi folosit pentru a identifica înregistrările după starea în care se află: inserate, modificate, şterse sau nemodificate.

Aplicațiile de tip **Briefcase** permit folosirea aplicației în timp ce este în mişcare. Tradițional în asemenea cazuri utilizatorii nu sunt conectați la baza de date, astfel că aplicația va trebui să conțină un modul pentru aceste cazuri, generând o colecție de fişiere Advanced Data Table Gram (ADTG) sau XML care să cuprindă copii ale datelor. Aplicația trebuie să verifice dacă este conectată la rețea, respectiv baza de date şi daca nu este, să folosească fişierele imagine pentru a obține date. În loc de UpdateBatch se utilizează SaveToFile, iar când se face reconectarea cu baza de date, se reîncarcă fişierele şi se face un UpdateBatch asupra bazei de date reale.

Datele extrase printr-un ADO dataset pot fi salvate într-un fişier pe disc pentru utilizare ulterioară pe aceeaşi sau altă stație de lucru. Datele pot salva în două formate: ADTG sau XML. Acestea sunt singurele formate suportate de ADO, dar nu sub toate versiunile.

Salvarea se face prin intermediul funcției SaveToFile care necesită doi parametrii: numele fişierului şi, opțional, formatul (ex. pfADTG sau pfXML). Dacă fişierul există este generată o excepție EOleException. Încărcarea datelor se face prin intermediul funcției LoadFromFile care necesită un singur parametru, numele fişierului. Dacă acesta nu există, de asemenea este generată o excepție.

Formatul ADTG este proprietar Microsoft fiind deci foarte eficient sub Windows. Daca se prefera se poate salva sub format XML. În acest caz trebuie luat însă în considerare faptul că ADO nu are un parser XML inclus, astfel trebuie instalat MSXML parser care se instalează cu Internet Explorer 5 şi mai nou sau de pe site-ul Microsoft. Totuşi există unele dezavantaje la folosirea formatului XML: salvarea şi încărcarea fişierelor este mai lentă decât formatul ADTG, iar fişierele generate sunt mai mari decât cele ADTG.

În cazul în care aplicația este single-tier şi nu de tip Briefcase se pot folosi proprietățile componentei ADODataSet CommandType cu valoarea cmdFile şi CommandText cu numele fişierului ca valoare. Astfel, nu va mai fi necesară folosirea metodei LoadFromFile manual.

Bibliografie

[Bor01] **Borland Software Corporation,** Borland Delphi 6 - Developer's Guide, Borland 2001

[MCS 03] **Marco Cantu Sybex,** Mastering Delphi 7, Borland 2003