Strategii de optimizare a performantelor unei aplicatii client/server

Prof.dr. Florin BICA, lect. Emanuela - Mariana CHICHEA
Universitatea din Craiova, Facultatea de Stiinte Economice
Catedra de Informatica Economica

O aplicatie client/server performanta presupune asigurarea unui echilibru între dezideratul obtinerii celei mai eficiente aplicatii pentru utilizatori, pe de o parte, si cel al asigurarii integritatii datelor, valorificarii superioare a hardware-lui, pe de alta parte.
Integritatea datelor este protejata prin utilizarea regulilor Visual FoxPro pentru o vedere externa sau off-line, a regulilor serverului si a declansatoarelor acestuia.
Performantele aplicatiei Visual FoxPro de tip client/server pot fi îmbunatatite prin urmatoarele cai: optimizarea utilizarii conexiunilor; marirea vitezei de obtinere a datelor; cresterea vitezei de rulare a interogarilor, vederilor si a formularelor; ameliorarea performantelor proceselor de actualizare si stergere.
Cuvinte cheie: *set de date, interogare parametrizata, vedere externa, vedere locala, conexiune, câmp marca de timp.*

1 Metodologia de dezvoltare a aplicatiilor client/server

Cu ajutorul instrumentelor Visual FoxPro pot fi create aplicatii client/server eficiente, în cadrul procesului de proiectare a unei asemenea aplicatii o importanta deosebita având proiectarea corecta a acesteia.

1.1. Utilizarea tehnicilor performante de proiectare

Proiectarea unei aplicatii client/server trebuie sa asigure pastrarea unui echilibru între dezideratul obtinerii celei mai rapide si eficiente aplicatii pentru utilizatori, pe de o parte, si cel al asigurarii integritatii datelor aplicatiei, valorificarii superioare a hardware-ului si asigurarii posibilitatii de dezvoltare ulterioara a aplicatiei, pe de alta parte; odata obtinut, acest echilibru ofera performante maxime în sistemele client/server.
Crearea în mediul Visual FoxPro a unei aplicatii client/server rapide si cu performante maxime are în vedere utilizarea noilor tehnici precum:
1. *Accesul eficient la date bazat pe seturi.*
Reducerea la minimum a cantitatii de date preluate de pe server impune renuntarea la tehnicile traditionale de navigare locala si utilizarea tehnicilor de acces la datele serverului bazate pe seturi, care fac posibila filtrarea cantitatii de date încarcate (în scopul maririi vitezei aplicatiei).
Accesarea datelor externe pe baza seturilor presupune selectarea dintr-un stoc urias a unui *set de date* prin intermediul instructiunilor SELECT - SQL.
2. *Interogarea parametrizata.*
Generarea unor interogari parametrizate care sa întoarca doar datele necesare este posibila gratie instructiunii SELECT bazate pe parametri prin care se poate transfera un set mic de date, dupa care se pot accesa noi înregistrari cu ajutorul functiei REQUERY(), ce solicita noul set de date.
Metodele de proiectare a unui sistem client/server pot fi multiple, alegerea uneia sau alteia dintre acestea permitând sau nu utilizarea în mod cât mai avantajos a tehnologiei client/server, astfel:
* *utilizarea unei strategii client/server neoptimizate* presupune aplicarea tehnicilor de parcurgere locala a datelor externe, prin indexarea tabelelor si folosirea comenzii SKIP pentru parcurgerea înregistrarilor.
Aceasta metoda se dovedeste a fi utila doar daca partea "una" a relatiei este locala iar cea "mai multe" este externa.
* *filtrarea partii "mai multe" a relatiei* limiteaza numarul înregistrarilor acesteia dar preia toate înregistrarile partii "una"

pentru a putea parcurge înregistrarile; eficienta acestei metode se resimte în situatia în care partea "una" a relatiei include un set mic de date.
* *filtrarea partii "una" a relatiei* presupune preluarea unui set mai mic de înregistrari, utilizând comanda SKIP pentru parcurgerea partii "una" a relatiei si functia REQUERY() pentru accesarea noilor date din partea "mai multe" a relatiei; aceasta metoda este indicata când partea "una" a relatiei, chiar dupa filtrare, ofera suficiente informatii pentru un set succesiv de interogari, înainte de a reinteroga serverul extern.
* *utilizarea cheii primare pentru accesul la relatia 1-n* presupune crearea unui formular care impune selectarea sau introducerea de catre utilizator a numarului de identificare a clientului, numar ce va servi apoi drept parametru pentru o vedere externa a fiecarei tabele relationate; se preteaza utilizarea acestei metode pentru accesarea aleatoare a relatiei 1-n prin intermediul unei valori oarecare a cheii primare.
În situatia introducerii de date externe în formular, este indicat a se include vederile în mediul de date al formularului: se atribuie proprietatii AutoOpenTables valoarea .F., indicându-se momentul în care se vor reîmprospata vederile cu datele externe, si se configureaza proprietatea ControlSource a casetelor de text sau a altor controale asociate datelor dupa apelarea metodei OpenTables, de obicei în cadrul evenimentului Init al formularului.
3. *Pastrarea datelor pe platforma de lucru optima* care depinde de modul în care aceste date sunt accesate si actualizate, relatia optima dintre elementele aplicatiei si platforma pe care trebuie plasate acestea fiind prezentata în tabelul urmator:

Elementul	**Amplasarea**	**Tipul**
TABELA	Local	Copii locale ale tabelelor de cautare de pe server; tabele mici, modificate la intervale mari de timp
	Extern	Tabele mari sau modificate la intervale scurte de timp
REGULA	Local	Reguli pentru vederile externe
	Extern	Reguli la nivelul rândurilor si coloanelor din tabelele de baza externe
PROCEDURA STOCATA	Local	Proceduri stocate în Visual FoxPro
	Extern	Proceduri stocate pe serverul back-end
TRANZACTIA	Local	Tranzactii Visual FoxPro
	Extern	Tranzactii de pe server
DECLAN§ATOR	Vederi locale	Fara declansatoare în vederi
	Extern	Declansatoare de pe server

În vederea crearii unei aplicatii client/server performante se poate combina utilizarea vederilor (pentru cea mai mare parte a nevoilor de gestionare a datelor) cu folosirea instructiunilor SQL de transfer (în vederea perfectionarii aplicatiei).
4. *Combinarea procedurilor VisualFoxPro si a procedurilor stocate extern* reprezinta cel mai performant model de creare a unei aplicatii Visual FoxPro client/server.
Principala metoda de dezvoltare a unei aplicatii client/server puternice o reprezinta vederile:
- vederile externe sunt proiectate în scopul selectarii de pe server doar a datelor necesare si actualizarii datelor externe;
- vederile locale permit crearea unui prototip local ce face posibila utilizarea Upsizing Wizard pentru transformarea vederilor locale în vederi externe.
Tehnologia SQL de transfer serveste pentru a efectua operatii specifice pe serverul extern (definirea datelor, crearea si executia procedurilor stocate pe server etc) cu ajutorul functiilor Visual FoxPro SQL de transfer care, comparativ cu vederile, per-

mit controlul si accesul suplimentar la server.

1.2. Dezvoltarea rapida si eficienta a aplicatiilor client/server

Procesul de dezvoltare a aplicatiilor client/server sub mediul Visual FoxPro presupune proiectarea si construirea unui prototip local al aplicatiei care va fi transformat si implementat treptat pentru o sursa de date externa, prin parcurgerea urmatoarelor etape:

- generarea unui prototip cu ajutorul vederilor: pot fi descoperite cu usurinta modificarile si îmbunatatirile ce trebuie aduse proiectului aplicatiei, acesta fiind perfectionat pentru esantioane mici de date;
- crearea unui prototip local cu ajutorul vederilor locale: un prototip local pentru o aplicatie client/server este o aplicatie Visual FoxPro functionala ce utilizeaza vederi locale pentru accesul la tabelele locale; desi vederile locale nu utilizeaza tabele dintr-o sursa de date externa ci tabele locale Visual FoxPro, datele locale sunt create astfel încât sa simuleze structura datelor de pe server;
- transformarea: creaza o baza de date pe serverul extern cu structura identica tabelelor, cu aceleasi date si chiar multe alte atribute ale bazei de date Visual FoxPro originale (în momentul transformarii, o aplicatie Visual FoxPro devine o aplicatie client/server);
- generarea de prototipuri cu ajutorul vederilor externe: salt peste etapa de transformare deoarece se utilizeaza în mod direct datele existente deja pe serverul extern si exista vederi externe pentru accesul la date.

1.3. Generarea unei aplicatii client/server care sa protejeze integritatea datelor

Obtinerea unei asemenea aplicatii devine posibila prin aplicarea combinata a regulilor de validare a datelor si a procedurilor stocate din Visual FoxPro si a regulilor de validare si procedurilor stocate din sursa de date.

Integritatea datelor poate fi pastrata prin:

- utilizarea regulilor Visual FoxPro pentru o vedere externa sau off-line: pot fi create reguli la nivel de câmp si de înregistrare pentru vederile externe si off-line în scopul validarii datelor introduse local, înainte ca acestea sa fie trasmise sursei de date externa;
- utilizarea regulilor serverului: servesc validarii datelor, rutina de tratare a erorilor putând apela functia AERROR() pentru a obtine informatii, inclusiv numarul mesajului de eroare, textul mesajului de eroare extern si identificatorul conexiunii asociate erorii;
- utilizarea declansatoarelor serverului: chiar daca nu pot fi create si pentru vederi ci doar pentru tabelele locale, declansatoarele Visual FoxPro pot fi folosite asupra sursei de date externe; declansatoarele serverului permit prelucrarea actualizarilor secundare mult mai eficient decât transmiterea de comenzi multiple catre serverul extern din aplicatia Visual FoxPro.

Sursele de date Visual FoxPro si majoritatea celor externe dispun de facilitati pentru tranzactii care previn pierderea de date.

2. Perfectionarea unei aplicatii client/server

Performantele unei aplicatii Visual FoxPro de tip client/server pot fi îmbunatatite prin urmatoarele cai:

a. *Optimizarea utilizarii conexiunilor*

Optimizarea unei conexiuni presupune crearea unui compromis între nevoia de performante ridicate si cerintele de resurse ale aplicatiei.

Utilizarea conexiunilor se poate realiza în doua moduri:

- *exclusiv*: dupa stabilirea conexiunii în aplicatie nu se vor naste conflicte pentru resurse;
- *partajat*: operatiile de manipulare a datelor din cadrul seturilor de rezultate care partajeaza aceeasi conexiune trebuie serializate iar proiectarea aplicatiei se va

face astfel încât la aparitia unui conflict sa se verifice daca acea conexiune nu este ocupata.
O conexiune poate fi inactiva iar proprietatea Idle Timeout controleaza intervalul de timp relativ la inactivitatea acestuia înaintea închiderii sale de catre Visual FoxPro. §i prin închiderea conexiunilor ce nu mai sunt utilizate de aplicatia client/server se poate obtine o crestere a performantelor acesteia.
b. *Marirea vitezei de obtinere a datelor*
Viteza de obtinere a datelor poate creste prin:
- gestionarea corespunzatoare a numarului de rânduri preluate în timpul extragerii progresive a datelor (metoda preluarii progresive a datelor din cursoarele vederii si cursoarele create asincron prin intermediul instructiunilor SQL de transfer); dezactivarea preluarii progresive a datelor permite preluarea rândurilor la cerere, cu ajutorul proprietatii FetchAsNeeded a bazei de date si a cursorului vederii, al carei efect consta în obtinerea mai eficienta a datelor din vederile externe sau vederi ce preiau seturi de rezultate foarte mari;
- controlul dimensiunii datelor preluate: prin configurarea proprietatii FetchSize a vederii se determina numarul de înregistrari preluate la un moment dat din serverul extern în cursorul local (valoarea implicita este 100);
- prelucrarea decalata a câmpurilor de tip memo: proprietatea FetchMemo a vederii sau a cursorului permite obtinerea celui mai rapid transfer al rândurilor si preluarea datelor din câmpuri de tip Memo sau General (de dimensiune foarte mare) doar daca este nevoie de acestea.
c. *Cresterea vitezei de rulare a interogarilor si vederilor*
Performantele vederilor si cele ale interogarilor pot fi îmbunatatite prin:
- adaugarea de indecsi tabelelor externe: indecsii grupati, creati automat pentru tabelele care au o cheie primara de catre SQL Server Upsizing Wizard, sunt cei mai performanti;
- optimizarea prelucrarilor locale si externe;
- optimizarea vederilor parametrizate, atribuind valoarea .T. proprietatii Prepared a vederii;
- optimizarea expresiilor parametrilor.
d. *Sporirea vitezei de rulare a formularelor*
Viteza de rulare a formularelor poate fi marita prin aplicarea urmatoarelor tehnici:
* încarcarea a mai putine date în formulare prin:
- solicitarea a cât mai putine înregistrari;
- utilizarea în vederile care stau la baza formularelor a cât mai putine câmpuri externe;
- utilizarea în setul de formulare a cât mai putine formulare care impun acces la vederile externe;
- utilizarea în numar restrâns a controalelor asociate cu datele externe;
* accelerarea încarcarii formularelor si reducerea încarcarii serverului, prin:
- pastrarea tabelelor care nu se modifica niciodata si care nu sunt prea mari în baza de date locala a aplicatiei;
- pastrarea tabelelor care sufera modificari la intervale mari de timp si a celor care se modifica ocazional, dar nu zilnic, atât pe server cât si în baza de date locala;
* afisarea câmpurilor numai la cerere, prin:
- atribuirea valorii .F. pentru proprietatea FetchMemo a vederii sau a cursorului, astfel încât câmpurile Memo sau General sa nu fie preluate pâna când nu sunt afisate pe ecran;
- atribuirea valorii .F. pentru proprietatea Visible a controalelor asociate câmpurilor de tip Memo sau General, ceea ce face ca utilizatorul sa poata decide pro sau contra vizualizarii continutului respectivelor controale;
- afisarea într-un formular principal a celor mai importante câmpuri si într-un alt formular a celorlalte câmpuri.
e. *Ameliorarea performantelor proceselor de actualizare si stergere***,** prin:
- adaugarea câmpurilor marca de timp (Timestamp) la tabelele externe, existenta unui astfel de câmp facând posibila utiliza-

rea optiunii de actualizare SQL Where-Type care economiseste timp de procesare prin compararea doar a doua câmpuri ale vederii (câmpul cheie si câmpul marca de timp) în scopul detectarii conflictelor de actualizare;
- excluderea câmpurilor de tip Memo din cauza WHERE a actualizarii prin atribuirea valorii .F. pentru proprietatea CompareMemo;
- utilizarea modului de tranzactionare manual care permite controlul asupra salvarii unui grup de tranzactii astfel încât serverul sa prelucreze rapid mai multe instructiuni, în defavoarea modului de tranzactionare automat; desi aceasta ultima metoda asigura cel mai bun control asupra fiecarei instructiuni de actualizare, introduce o supraîncarcare, inconvenient ce poate fi înlaturat prin marirea valorii proprietatii BatchUpdateCount a vederii sau a cursorului;
- utilizarea procedurilor stocate pe server care, fiind precompilate, se executa foarte repede;
- crearea loturilor de comenzi pentru actualizari astfel încât sa fie asigurata o gestionare mai eficienta a acestora de catre retea si server, prin marirea valorii proprietatii BatchUpdateCount a vederii (valoarea implicita este 1).

Alaturi de aceste strategii care vizeaza optimizarea performantelor unei aplicatii client/server sub mediul Visual FoxPro, configuratia calculatorului si cerintele aplicatiei contribuie în mod decisiv la stabilirea performantelor reale ale acesteia.

Bibliografie

1. D. Lowe - *Tehnologia client/server pentru toti*, Editura Teora, Bucuresti, 1996
2. B. Pârv, Al. Vancea - *Fundamentele limbajelor de programare*, Editura Albastra, Cluj-Napoca, 1996
3. M. Antonovich - *Utilizarea Visual FoxPro 3 pentru Windows*, Editura Teora, Bucuresti, 1997
4. O. Bâsca - *Baze de date*, Editura All, Bucuresti, 1997
5. R. Dollinger - *Baze de date si gestiunea tranzactiilor*, Editura Albastra, Cluj-Napoca, 1997
6. A. Vulpe, D. Bucerzan - *Lectii de FoxPro*, Editura Albastra, Cluj-Napoca, 1997
7. Fl. Bica, E. Chichea - *Baze de date FoxPro. Organizare si exploatare*, Editura Universitaria, Craiova, 1998
8. P. Petrus - *Visual FoxPro 5.0*, Editura Promedia Plus, Cluj-Napoca, 1998
9. Fl. Bica, E. Chichea - *FoxPro: Programare interactiva*, Editura Universitaria, Craiova, 1999
10. G. Dinea, M. Dinea - *Bazele Visual FoxPro 5.0*, Editura Teora, Bucuresti, 1999
11. Microsoft - *Visual FoxPro 6.0* - Ghidul Programatorului, Editura Teora, Bucuresti, 2000