Optimizing Queries in Distributed Systems

Prof.dr. Ion LUNGU, Anca Georgiana FODOR
Catedra de Informatică Economică, A.S.E. Bucureşti

This research presents the main elements of query optimizations in distributed systems. First, data architecture according with system level architecture in a distributed environment is presented. Then the architecture of a distributed database management system (DDBMS) is described on conceptual level followed by the presentation of the distributed query execution steps on these information systems. The research ends with presentation of some aspects of distributed database query optimization and strategies used for that.
Keywords: *distributed system, distributed system architecture, distributed query, distributed query optimization.*

Sisteme de baze de date distribuite

Apariția sistemelor distribuite a fost determinată de modul de desfăşurare a activității companiilor cu sedii aflate la distanțe considerabile unele de altele, uneori chiar pe continente diferite, structura organizațională promovând un mod de lucru descentralizat.
Sistemele distribuite ajută la rezolvarea problemei "insulelor informaționale", apărute ca urmare a separării geografice dar şi datorită diversității arhitecturilor calculatoarelor şi a protocoalelor de comunicație.

În prezent, sistemele de baze de date distribuite sunt organizate ca sisteme software multifir, conform [SMBL02]. Arhitectura generică pentru sistemele de baze de date distribuite este arhitectura pe trei niveluri fiind alcătuită din componenta de gestiune a bazei de date, componenta de gestiune a proceselor şi aplicațiilor şi respectiv componenta de interfață cu utilizatorii. Similar acestei arhitecturi software cu trei componente, există **arhitectura datelor**, pe trei niveluri, prezentată în figura 1.

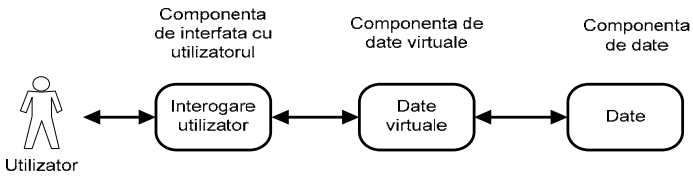

Figura 1. Arhitectura datelor pe trei niveluri

Componenta de date este alcătuită din datele stocate în sistemul distribuit care sunt disponibile în infrastructura rețelei de calculatoare. *Componenta de date virtuale* asigură transparența distribuției datelor, oferind utilizatorului accesul unitar la date. *Componenta interfeței cu utilizatorul* permite acestuia să formuleze interogări asupra uneia sau mai multor view-uri.
Responsabilitățile funcționale din arhitectura datelor prezentată mai sus revin unor sisteme de calcul, aşa cum se vede în figura 2. Componenta de date este reprezentată de unul sau mai multe sisteme de gestiune a bazelor de date (*SGBD*) care funcționează pe mai multe *servere de baze de date*. Datele virtuale sunt gestionate de componente software ce rulează pe un *server de aplicații*. *Calculatorul client* este punctul de acces al utilizatorului la sistem şi modelează componenta de interfață cu utilizatorul din arhitectura datelor. Funcția acestuia este doar de a prelua interogările utilizatorului şi a-i prezenta acestuia rezultatul cererilor sale. În unele cazuri, părți din procesarea interogărilor pot fi direcționate către calculatorul client, dar în general, acesta are mai mult rolul de comunicare decât de prelucrare.
Toate cele trei componente manipulează date. Fiecare componentă accesează datele de la componenta anterioară şi le transferă componentei următoare. În final utilizatorul va

primi datele prin intermediul componentei "Interogare utilizator" care funcţionează pe calculatorul client.
În cazul sistemelor în care componenta de date virtuale lipseşte, distribuţia datelor nu este ascunsă utilizatorului, şi acesta trebuie să formuleze interogări direct SGBD-urilor gestionându-şi singur distribuţia datelor.

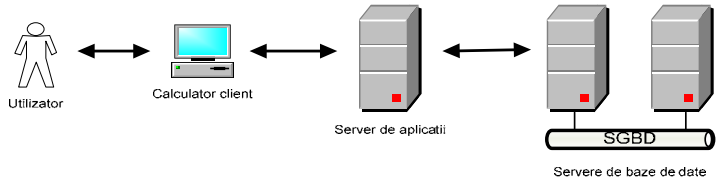

Figura 2. Arhitectura de sistem pe trei niveluri

Arhitectura de referinţă a unui SGBDD
Sistemele de gestiune a bazelor de date distribuite (*SGBDD*) promovează o arhitectură multinivel pentru a obţine transparenţa distribuţiei, aşa cum se poate vedea şi în figura 3. Arhitectura SGBDD de tip ANSI/SPARC este o arhitectură de referinţă pentru bazele de date distribuite şi prezintă conceptual o bază de date distribuită (*BDD)*. Această arhitectură nu este implementată explicit în toate sistemele de baze de date distribuite.

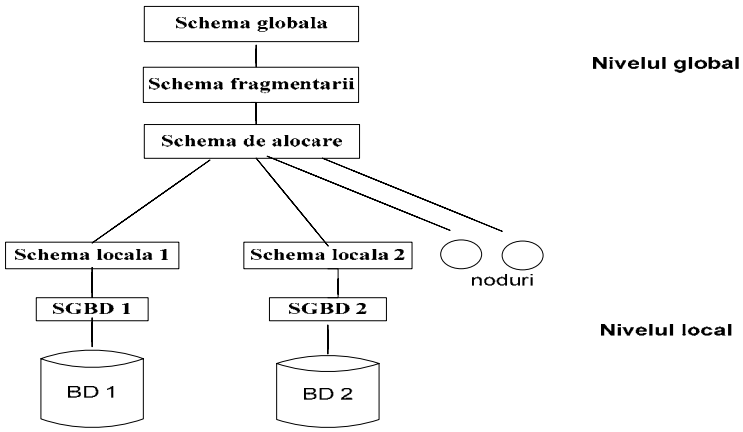

Figura 3 Arhitectura de referinta a SGBDD

Într-un SGBDD, datele sunt organizate pe două nivele: nivelul global şi nivelul local. **Nivelul global** permite integrarea bazelor de date locale într-o bază de date globală prin intermediul schemei globale, schemei de fragmentare şi schemei de alocare. *Schema globală* defineşte toate datele conţinute în BDD şi este descrisă de un set de relaţii globale. Fiecare relaţie globală poate fi împărţită în mai multe părţi disjuncte numite **fragmente**. Corespondenţa între relaţia globală şi fragmente este definită în *schema de fragmentare*. Fragmentele sunt porţiuni logice ale relaţiilor globale care pot fi alocate fizic pe una sau mai multe staţii ale reţelei. *Schema de alocare* defineşte nodurile unde sunt alocate fragmentele. **Nivelul local** permite tratarea fiecărei baze de date locale ca o bază de date centralizată. La acest nivel *schema locală* – care depinde de tipul SGBD local - realizează corespondenţa între imaginea fizică a relaţiilor globale la nod şi obiectele manipulate de SGBD-ul local.

Execuţia cererilor în sistemele de baze de date distribuite
Conform [OSVA95], în sistemele distribuite, *obiectivul execuţiei cererilor* este de a transforma o cerere globală, adresată bazei de date văzută ca un întreg de către utilizatorii ei, în comenzi de manipulare a datelor de nivel scăzut, adresate bazelor de date locale, aflate la noduri, folosind o strategie de execuţie eficientă.
Execuţia unei cereri distribuite este compusă din mai multe activităţi, corespunzătoare etapelor generice ale procesării cererii. Conform [OSVA95], aceste etape se desfăşoară ca în figura 4.

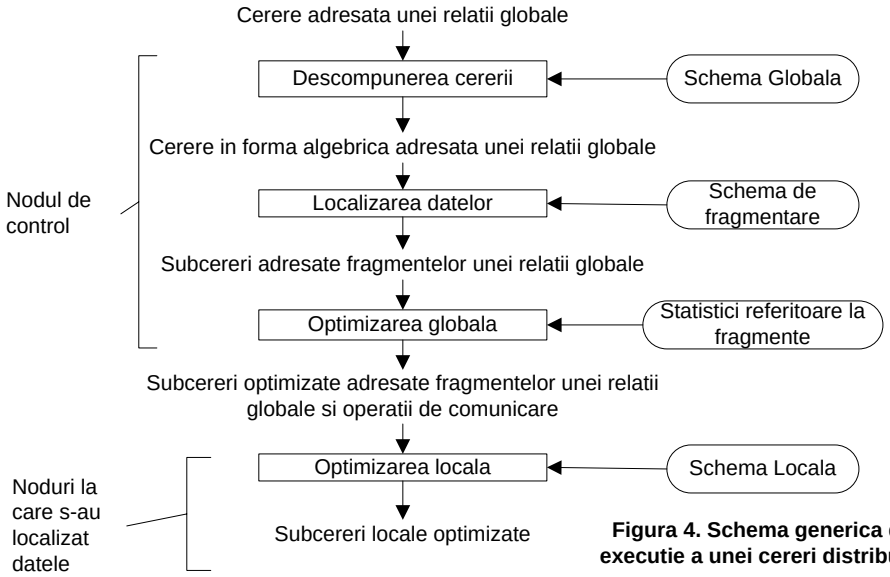

Figura 4. Schema generica de executie a unei cereri distribuite

Execuția unei cereri porneşte de la cererea globală exprimată folosind operatorii de calcul relațional. Această cerere este adresată unor relații globale, distribuirea datelor fiind invizibilă la acest nivel. Ea este descompusă în subcereri exprimate cu ajutorul operatorilor algebrei relaționale. Aceste subcereri se transformă din subcereri globale în cereri adresate fragmentelor relațiilor globale, folosind informații despre schema de fragmentare. Optimizarea globală constă în găsirea celei mai bune ordonări a operațiilor în cadrul subcererilor adresate fragmentelor, incluzând şi operațiunile de comunicație care minimizează o funcție de cost. Optimizarea locală a fiecărei subcereri la un anume nod se rafinează folosind informații din schema locală.

Aceste patru etape realizează descompunerea cererii globale într-o secvență de operațiuni locale optimizate, fiecare din acestea acționând asupra unei baze de date locale. Primele trei se desfăşoară la un nod de control, care deține informații despre schema globală, ultima etapă fiind executată la nodurile unde sunt localizate datele necesare procesării cererii.

Factori care influențează optimizarea cererilor distribuite

O etapă importantă în execuția cererilor distribuite o reprezintă optimizarea execuției acestora. Deşi există mai multe variante de transformare a aceleiaşi cereri globale, este esențial să reținem ca soluție acea variantă care optimizează (minimizează) consumul de resurse cu prețul unui efort rezonabil.

Interogarea rezultată în urma etapei de localizare a datelor este o interogare adresată fragmentelor exprimată în algebra relațională şi are o formă ce poate fi executată, elementele neesențiale fiind deja eliminate.

Este totuşi necesară o optimizare a acestei forme a cererii ținând cont de dimensiunile fragmentelor implicate în execuție, de caracteristicile rețelei de comunicație peste care funcționează sistemul distribuit, precum şi de resursele fizice ale acestuia (capacitate de procesare, memorie disponibilă şi spațiu pe discuri).

Optimizarea poate fi privită ca activitatea prin care se încearcă obținerea celei mai bune soluții posibile în condițiile de mediu date. Spun posibile, deoarece nu întotdeauna se poate obține soluția "cea mai bună", în valoare absolută, din motive lesne de înțeles, şi anume, sau ar necesita un timp prea mare de căutare în spațiul soluțiilor, sau ar bloca prea multe resurse, comparativ cu o soluție apropiată de "cea mai bună", dar care s-ar putea obține cu eforturi vizibil mai mici .

Problema optimizării execuției unei interogări distribuite are ca *obiectiv major* obținerea unui timp de răspuns al sistemului cât mai mic în condițiile minimizării costului total de execuție.

Costul total de execuție al unei interogări într-un sistem distribuit, conform [OSVA95], se compune din costul de procesare, costul de accesare a discurilor şi costul comunicației

între noduri. Costul comunicației se compune din costul trimitere şi primire mesaje între noduri şi respectiv costul de transfer date între noduri. Costurile de comunicație depind în mare măsură de tipul rețelei de comunicație. În cazul unei rețele LAN, putem considera costul unitar de transfer de date identic între oricare două noduri. În cazul unei rețele WAN, costurile unitare de transfer date între noduri sunt diferite, în funcție de distanța între noduri care poate varia semnificativ de la o pereche de noduri la alta. De asemenea, în cazul WAN, costurile de procesare şi respectiv costurile de accesare a discurilor sunt nesemnificative față de cele de comunicație şi deci se pot neglija, având funcția de cost total compusă doar din costul de transfer date şi costul de transmitere/primire de mesaje. Formula costului total se poate scrie astfel [OSVA95]:

Cost total*=C_{CPU} * *nr_instrucțiuni*+$C_{I/O}$ * *nr_I/O* + C_{MSG} * *nr_mesaje* + C_{TR} * *cantitate_date

Timpul de răspuns al sistemului distribuit se calculează din momentul lansării în execuție a interogării până la primirea răspunsului din partea sistemului. O influență semnificativă în minimizarea timpului de răspuns o au procesarea paralelă şi respectiv comunicația paralelă. Cu cât creşte execuția şi respectiv comunicarea în paralel, cu atât timpul de răspuns va fi mai mic. Şi în cazul timpului de răspuns este foarte important tipul rețelei WAN sau LAN, deoarece, timpul de execuție a cererii, precum şi timpul necesar accesului la disc devin nesemnificative față de timpul necesar transferului de date şi cel al transmiterii de mesaje, în cazul rețelelor WAN, spre deosebire de rețelele LAN. Formula timpului de răspuns al sistemului distribuit se scrie, conform [OSVA95]:

Timp răsp* = T_{CPU} * *nr_instrucțiuni* + $T_{I/O}$ * *nr_I/O* + T_{MSG} * *nr_mesaje* + T_{TR} * *cantitate_date

După cum am precizat, minimizarea timpului de răspuns se realizează prin creşterea gradului de paralelism al execuției unei interogări distribuite. Aceasta nu implică neapărat că şi costul total va fi minimizat. Din contră, costul total poate creşte încercând să sporim gradul de paralelism al execuției şi transmisiei. Pe de altă parte, minimizarea costului total implică îmbunătățirea utilizării resurselor, în detrimentul timpului de răspuns, care se va mări. În practică este de dorit un compromis între cele două obiective.

Într-unul din studiile de specialitate [KAMO94] se descriu două *strategii de optimizare a execuției interogărilor* atunci când sunt necesare date de la mai multe noduri. Acestea strategii sunt: *query-site* şi *move-small*.

Strategia *query-site* spune că dacă avem o interogare care foloseşte date ce se regăsesc în mai multe fragmente la noduri diferite, cel mai "ieftin" ar fi să transferăm toate acele fragmente la nodul de unde s-a inițiat interogarea şi să o executăm acolo.

Strategia *move-small* spune că dacă o operație binară implică două fragmente ce se află la noduri diferite, atunci transferăm fragmentul mai mic la nodul unde este localizat fragmentul mai mare.

Este vorba desigur de rezultatele intermediare care s-au obținut în urma operațiunilor unare (selecție, proiecție) şi care urmează să participe la operațiile de "join" sau reuniune, mari consumatoare de resurse. Este important să ținem cont de aceste strategii de execuție pentru a minimiza cantitatea de date transferate între noduri, şi implicit, costul de comunicație, care este o parte semnificativă a costului total, în special în rețele de tip WAN.

Componentele optimizării cererilor distribuite, conform [MULL96], sunt metoda de acces la date, criteriile de uniune folosite şi costurile de comunicație.

Majoritatea SGBD-urilor permit accesarea tabelelor în două moduri: secvențial sau indexat. Cea mai potrivită metodă de acces la date poate fi determinată de contextul cererii. Este posibil ca aceasta să returneze 90% din tuplurile tabelei, caz în care vom folosi accesul secvențial. În cazul în care cererea va returna 10% din tupluri, folosirea unui index ar creşte viteza de răspuns a sistemului.

Dacă interogarea accesează mai multe tabele cărora li se va aplica operatorul de uniune, este determinant modul în care se realizează aceasta. Optimizarea cererii trebuie să ia în

considerare ordinea în care facem uniunea tabelelor şi numărul tuplurilor care iau parte la "join". De asemenea, este important cu care nod al reţelei începem operaţiunea de "join". Dacă datele de la mai multe noduri participă la "join" pentru a executa o interogare, trebuie luat în calcul şi costul transmiterii rezultatelor intermediare între noduri.

Optimizarea unei interogări se poate realiza de către SGBD sau de către aplicaţiile utilizator. Optimizarea se realizează de către SGBD dacă acesta conţine algoritmi interni de optimizare a fiecărei interogări. Deşi, o optimizare a SGBD ar fi de dorit, nici un modul de optimizare al SGBD nu este întotdeauna suficient de complet pentru a determina cum este cel mai bine să facă operaţiunea de "join" între două tabele stocate la noduri diferite.

În cazul absenţei unei optimizări oferite de SGBD, programatorul trebuie să optimizeze fiecare interogare scriind cod pentru operaţiunile selecţie şi uniune realizate între noduri pentru fiecare program de aplicaţie care necesită aşa ceva. În acest caz, programatorul trebuie să ţină seama de câţiva factori: dimensiunea tabelelor; localizarea tabelelor la noduri; disponibilitatea indecşilor; eficienţa reţelei; necesitatea posibilităţii de scriere într-un limbaj procedural a unor secvenţe de cod ce nu pot fi scrise folosind doar SQL (un limbaj neprocedural); disponibilitatea datelor (fragmente, replici, snapshot-uri); să folosească, pe cât posibil, cod reutilizabil, pentru ca modificarea acestora să fie facilă.

În plus faţă de aspectele menţionate anterior, se mai pot enumera şi alte considerente care pot duce la creşterea complexităţii problemei şi care, cu siguranţă, trebuie luate în calcul la optimizarea interogărilor distribuite. Trebuie amintit de aspecte legate de securitatea sistemului şi implicaţiile autorizării accesului la date: cine ce anume date poate accesa şi la ce nod trebuie implementate rutinele de autorizare. Intercalarea rutinelor de autorizare în fluxul de execuţie al operaţiunilor în sistem duce la creşterea timpului de răspuns a acestuia la interogări.

Un alt aspect este legat de disponibilitatea resurselor. La un moment dat este posibil, şi cu siguranţă se întâmplă în decursul timpului ca unul din noduri, să nu fie disponibil pentru interogare, sau chiar reţeaua de comunicaţie să nu fie utilizabilă, Acest lucru creşte timpul de răspuns şi implicit costul total, sau, chiar mai rău, poate face imposibilă execuţia interogării în cazul în care fragmentul ce conţine datele dorite nu este replicat la un alt nod disponibil în acel moment.

Constrângerile de integritate declarate la nivelul bazei de date globale ajută la optimizarea interogărilor adresate sistemului. Se pot astfel elimina din arborele cererii anumite ramuri ce nu se pot întâlni datorită acestor constrângeri de integritate.

Procesul de optimizare a interogărilor distribuite este în strânsă dependenţă de implementarea şi modul de utilizare a reţelei de comunicaţie. Traficul în reţea poate varia de la o zi la alta, chiar de la o oră la alta, ştiut fiind faptul că există perioade din zi cu vârfuri de sarcină pentru reţea şi perioade cu trafic foarte scăzut. De altfel, pe măsură ce reţeaua suferă modificări, de orice tip, soluţia de optimizare trebuie revăzută, şi probabil chiar modificată, în cazul în care apar noi posibilităţi mai rapide de comunicare între noduri.

Am enumerat câteva aspecte cu impact semnificativ asupra timpului de răspuns şi implicit al costului total de execuţie al unei interogări distribuite. De asemenea, cu cât creşte numărul de factori care influenţează procesul de execuţie al cererilor, cu atât optimizarea acestuia devine o activitate deosebit de complexă.

Concluzii

Optimizarea cererilor în sistemele distribuite este o activitate complexă care depinde de mulţi factori. În parte ea este realizată de SGBD, dar există situaţii în care aplicaţiile utilizator trebuie să conţină şi algoritmi de optimizare a cererilor.

Dacă fragmentarea bazei de date este realizată corect, o mare parte din cererile utilizatorilor se vor executa la nivel local şi astfel execuţia cererii va depinde în mare măsură de indecşii pe care i-am creat în deplină concordanţă cu predicatele cererii. Pe de altă parte, politica de securitate a sistemului va restricţiona accesul utilizatorilor la o plajă de înre-

gistrări bine precizată, asupra cărora acționează şi un set de constrângeri de integritate.
Cu toate că putem avea o bază de date bine proiectată astfel încât proporția prelucrărilor locale să fie foarte mare şi trierea cererilor de către rutinele de securitate să fie drastică, totuşi, în decursul exploatării sistemului distribuit se va dori execuția unor cereri mai complexe care să necesite date de la mai multe noduri. Totodată, în funcție de frecvența de execuție a cererilor în sistemul distribuit, se va justifica o căutare completă în spațiul soluțiilor obținându-se optimul absolut sau se va opta pentru o soluție apropiată de optim găsită cu ajutorul algoritmilor euristici. În aceste cazuri va trebui să ținem seama de aspectele de optimizare menționate în acest articol.

Bibliografie

[BERN81], P.A.Bernstein, N. Goodman, E. Wong, C.L. Reeve , J.B. Rothnie –*Jr. Query Processing in a System for Distributed Databases (SDD-1)*- ACM Trans. Database Syst., 1981.

[EPST78], R. Epstein, M. Stonebraker, E.Wong – *Query Processing in a Distributed Relational Database System*- In Proc. ACM SIGMOD Int. Conf. on Management of Data, Austin, 1978.

[KAMO94], K. Karlaplem, N. Moon Pun - Query Driven Data Allocation Algorithms for Distributed Database Systems - RGC CERG, 1994.

[KOSS00], D. Kossman – *State of the art in distributed query processing* – ACM Computing Surveys, decembrie 2000.

[LUJA02], G. Lumpkin, H. Jakobsson – *Query Optimization in Oracle9i* – An Oracle White Paper, februarie 2002.

[LUBO95], I. Lungu, C. Bodea, G. Bădescu, C. Ioniță – *Baze de date, Organizare, Proiectare şi implementare* - Ed. ALL, 1995.

[MULL96], C.S.Mullins – *Distributed query optimization* – Technical Support Magazine, iulie 1996.

[NAKA94], S.B. Navache, K. Karlapalem, M. Ra – *A Mixed Fragmentation Methodology For Initial Distributed Database Design* – College of Computing, Georgia Institute of Technology, 1994.

[OSVA95], M. T. Oszu, P. Valduriez – *Principles of distributed database systems* – Prentice-Hall International, 1995.

[SMBL02], M. Smilijanic, H. Blanken, M. van Keulen, W. Jonker –*Distributed XML Database Systems* – Twente University, Database Group, octombrie 2002.

[ULLM82], J. D. Ullman - *Principles of database systems* - Rockville, Computer Science Press, 1982.