Optimizarea interogarilor în bazele de date relationale

Prof.dr. Ion LUNGU, lect. Ileana TANASE
Catedra de Informatica Economica, A.S.E. Bucuresti

Optimizarea interogarilor reprezinta o facilitate importanta a bazelor de date si reflecta eficienta si performanta acestora. Tehnicile de optimizare au la baza echivalenta dintre algebra relationala si calculul relational, precum si proprietatile operatorilor algebrici.
Cuvinte cheie: *baze de date relationale, graf de strategii, tehnici de optimizare.*

Etapele optimizarii cererilor de date

Ne propunem sa tratam optimizarea cererilor de date dintr-o baza de date relationala. O astfel de optimizare se realizeaza prin parcurgerea urmatoarelor etape:

1. Exprimarea cererilor cu ajutorul algebrei relationale (expresii algebrice relationale).
2. Obtinerea unor expresii echivalente celor initiale, dar care sa poata fi executate cât mai eficient.

Transformarile echivalente implica doua aspecte:

a) *Logic* - transformarile sunt bazate pe proprietatile operatiilor de selectie în fata operatiilor de join (deplasarea operatiilor de proiectie înaintea operatiilor de join, deplasarea selectiilor în fata proiectiilor, combinarea selectiilor multiple);

b) *Semantic* - transformarile sunt bazate pe proprietatile atributelor.

Exemplu: Presupunem ca relatia la care ne referim este:

EMPLOYEE (Emp # , Dept, Salary) si daca Dept=liqeur => Salary < 30000

Atunci:

σ(Salary<100000>$\wedge$(Dept=liqeur)
(EMPLOYEE)$\equiv\sigma$(EMPLOYEE)(Salary
<30000> $\wedge$(Dept =liqeur)

Expresia echivalenta poate fi implementata eficient daca sunt folositi

3. Reprezentarea expresiei obtinute folosind grafurile de strategii.
4. Pentru fiecare program putem aproxima costul de executie, bazându-ne pe estimari ale parametrilor relevanti (de exemplu, dimensiunea tablourilor). În final vom putea selecta acel program care are costul de executie cel mai mic.

Graful de strategii

Consideram problema implementarii expresiei: M $\{T_1, T_2,...,T_k)$

1. Primul pas este de a reprezenta expresia printr-un graf G (*graf de strategii*), care contine noduri corespunzatoare fiecarui tabel T_i, $i = \overline{1, k}$ si noduri corespunzatoare fiecarui operator k=join.
2. Pasul urmator ar putea fi considerarea tuturor programelor pentru fiecare arbore obtinut. Pentru fiecare program putem apro-xima costul de executie si, în final, vom putea selecta programul cu costul de exe-cutie cel mai mic.

Optimizarea pentru $\{T_{1,...}T_k\}$ implica un efort enorm. La primul pas am utilizat termenul de graf de strategii, care furnizeaza un excelent cadru pentru studierea tehnicilor de cautare în procesul de optimizare al inte-rogarilor si reprezinta unelte cu care descri-em si studiem comportamentul optimizarii.

În literatura de specialitate exista mai multe reprezentari ale grafurilor de interogari: Graefe

and De Witt (1987), Jarke and Kock (1984), Joussefi and Wong (1970). Grafurile de strategie sunt un tip de grafuri de intero-gari, propus prima data de Runer si Rosen-thal (1984).

Reprezentarea grafurilor de strategii este similara oricarui graf. Diferenta fata de celelalte grafuri consta în faptul ca nodurile sunt de doua feluri: *patrate* pentru tabele si *cercuri* pentru operatori, iar arcele au urma-toarea semnificatie:

- daca tabelul n_1 este intrare pentru operatorul n_2, atunci $n_1 \rightarrow n_2$;
- daca tabelul n_3 rezulta prin aplicarea operatorului n_2, atunci $n_2 \rightarrow n_3$.

Graful are un nod radacina (nod rezultat), fiind un nod tabel care reprezinta rezultatul interogarii. Nodurile la care nu sosesc arcuri se numesc noduri de baza, iar celelalte sunt noduri intermediare.

Pentru o mai buna întelegere vom construi graful de strategii al interogarii $\Pi_{<B,C>}$ $(\sigma_{A=v}(R_1) \bowtie R_2)$, reprezentat în figura 1.

Graful din figura 1 reprezinta o singura strategie, deoarece fiecare nod tabel are cel mult o intrare. Daca permitem existenta mai multor intrari, atunci graful de strategii G poate reprezenta complet o multitudine de strategii candidat pentru interogarea propusa.

În figura 2 se prezinta un graf de strategii reprezentând doua strategii.

Este util sa identificam fiecare strategie din cadrul unui graf care reprezinta mai multe strategii. În acest scop vom defini notiunile de:

- *strategie singura*, care este strategia repre-zentata printr-un graf initial în care fiecare nod tabel are cel mult o intrare.
- *strategie singura inclusa* într-un graf de strategii G, care este un subgraf G' a lui G care îndeplineste conditiile:

a) G si G' au acelasi nod rezultat;

b) nodurile de baza ale lui G' formeaza o submultime a nodurilor de baza a lui G;

c) fiecare nod operator **m** care apartine lui G' are toate intrarile tot în G' (toate intrarile posibile din G);

d) fiecare nod tabel a lui G' are cel mult o intrare.

În figura 2, nodurile A, B, C, E, F, I, J, Q induc o strategie inclusa.

Atunci când se doreste sa se accentueze ca un subgraf calculeaza un rezultat interme-diar, se utilizeaza termenul de *substrategie*, iar atunci când se accentueaza ca un subgraf calculeaza un rezultat final se utilizeaza termenul de *strategie completa*.

În figura 2, subgraful compus din nodurile A, B, C, E, F si I este o substrategie inclusa pentru calculul rezultatului intermediar σ (R1 R2).

Pentru a ilustra conceptele enuntate, consi-deram graful de strategie din figura 3:

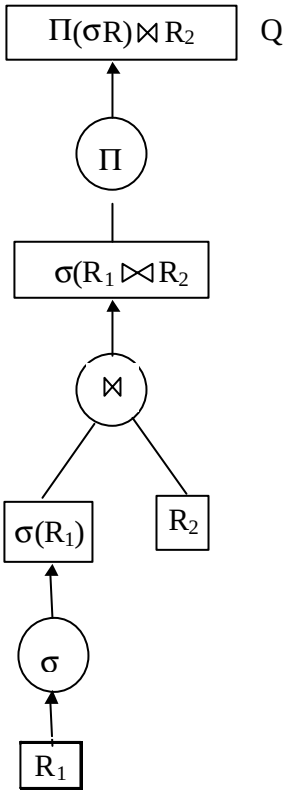

Figura 1

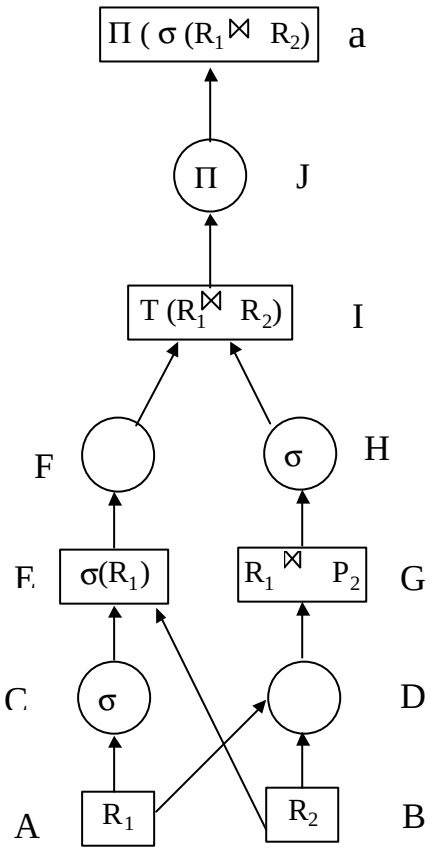

Figura 2

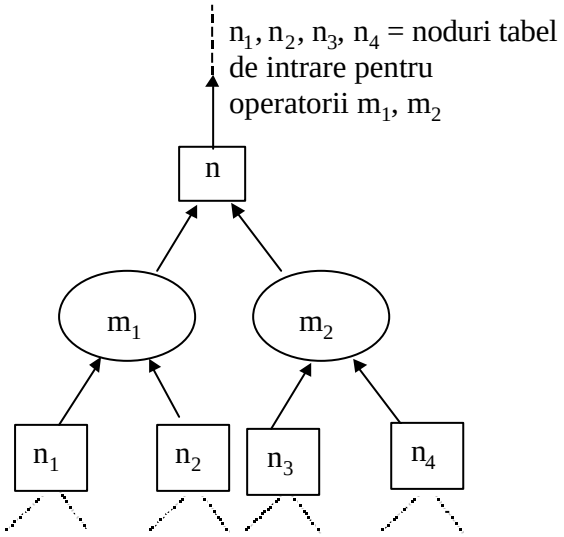

Figura 3

Nodurile m_1 si m_2 sunt ultimele operatii dintr-o colectie de substrategie pentru produ-cerea rezultatului intermediar **n**. Pentru fie-care nod n_i $\left(i=\overline{1,4}\right)$ graful contine N_i strate-gii pentru obtinerea lui n_i. În consecinta, graful contine $(N_1*N_2)+(N_3*N_4)$ substrategii pentru obtinerea lui **n**. Rezultatul procesului de optimizare a interogarilor este selectia u-nei singure strategii dintr-un graf de strategii.

Costul strategiilor

O strategie este compusa dintr-o colectie de algoritmi care implementeaza operatorii al-gebrici. Strategia aleasa pentru realizarea optimizarii va trebui sa aiba costul cel mai mic. Costul unui nod operator se defineste ca fiind costul aplicarii la intrare a operatorilor specificati în algoritmi, iar costul unei strategii reprezinta suma costurilor nodurilor operator. Fie graful din figura 4.

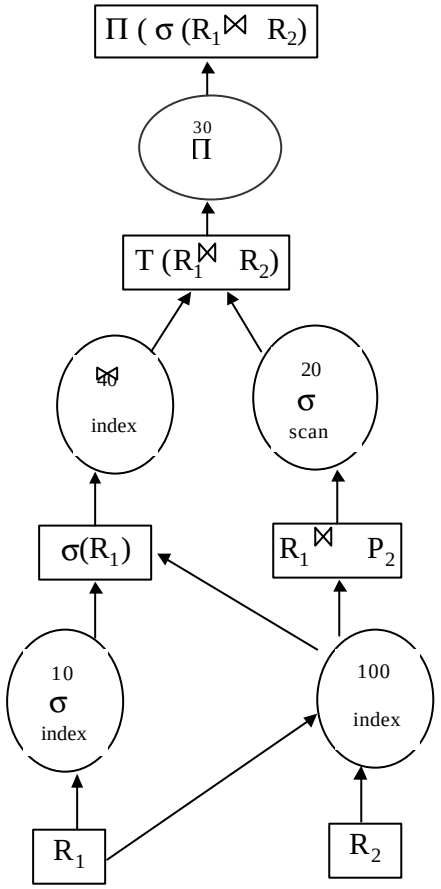

Figura 4

În graful din figura 4 am inclus în fiecare nod operator un cost estimat în termenii câtorva unitati standard (de exemplu, acce-sarea blocului). Pornind de la aceasta estimare putem calcula costul fiecarei stra-tegii incluse în G. Strategia care calculeaza Join-ul înaintea lui Select costa 150, în timp ce strategia care calculeaza Select înaintea lui Join costa 80.

Calea aleasa pentru cautare în graful de strategii consta în evaluarea fiecarei strategii incluse si compararea costurilor. Din nefericire, aceasta evaluare poate consuma foarte mult timp si, deseori, este imposibil de realizat acest lucru. Nerealizarea este cauza-ta, în general, de numarul foarte mare al strategiilor prezente într-un graf. De obicei, numarul acestora creste exponential în functie de numarul nodurilor si al arcelor. Chiar daca un graf de strategii este obtinut prin aplicarea unui numar relativ mic de transformari, el ar putea contine un numar foarte mare de programe, facând imposibila evaluarea fiecaruia separat. Pentru optimiza-rea strategiilor incluse exista o varietate de tehnici pentru cautarea unui graf de strategii.

Tehnici de cautare

1) Tehnici de legatura. Presupunem ca am evaluat o strategie completa S al carui cost este Cost(S). Se evalueaza si celelalte strate-gii care calculeaza un nod intermediar **n**. Daca costul acestora este mai mare decât Cost(S), atunci este evident ca o strategie optima nu poate calcula **n** si deci se poate simplifica. Simplificare este modelata prin taierea tuturor muchiilor care pleaca de la fiecare nod operator care ia pe **n** ca intrare. O astfel de simplificare este sigura (nu elimina strategia optima), dar este dificil sa obtinem o strategie completa buna care sa joace rolul lui S. Discutia urmatoare sugereaza folosirea euristicilor pentru obtinerea unei strategii bune (si într-un timp cât mai mic), care poate fi apoi utilizata în comparatiile de legatura. De fiecare data, cautarea genereaza o strate-gie completa, care este mai buna decât cea mai buna strategie curenta. Aceasta noua strategie poate fi folosita în comparatiile ulterioare.

2) Tehnici de programare dinamica. În timp ce costurile de legatura compara costul unei strategii complete cu costurile substrategii-lor, algoritmii de programare dinamica com-para între ei costurile substrategiilor care calculeaza un rezultat intermediar **n**. Deza-vantajul este acela ca o substrategie simpli-ficata ar putea fi de fapt o parte a unei stra-tegii optime si simplificarea va determina selectarea unei strategii suboptime.

În ciuda acestei limitari programarea dinami-ca ramâne una dintre cele mai puternice tehnici de cautare disponibile.

3) Tehnici de cautare euristice. Tehnicile euristice pot accelera cautarea, dar nu sunt garantate sa permita gasirea unei strategii optime. De exemplu, putem extinde tehnica de legatura care sa se aplice de câte ori gasim un cost al celei mai bune strategii pentru calcularea unui nod tabel intermediar **n** de 90% (sau mai mult) din cea mai buna strategie actuala. Abordarea are un caracter intuitiv si nu exclude eliminarea strategiei optime. Alte tipuri de euristici sunt gândite sa determine în ce ordine strategiile ar trebui evaluate. De exemplu, sistemul INGRES foloseste o euristica care coreleaza timpul de cautare pentru o strategie cu estimarea complexitatii proprii.

Algoritmi pentru gasirea strategiei optime

În continuare, se prezinta un algoritm de programare dinamica, pentru gasirea efici-enta a strategiei optime într-un graf, bazat pe cautarea bottom-up (figura 5).

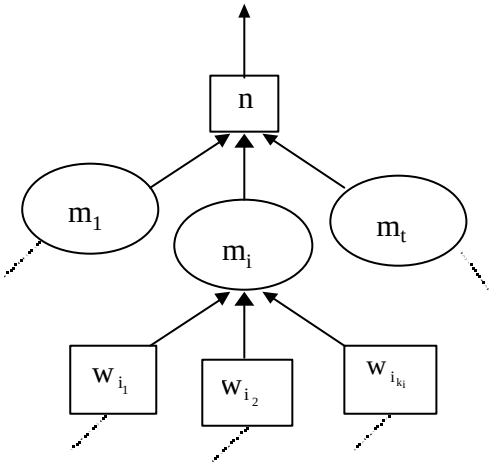

Figura 5

for (fiecare nod tabel **n** a lui G, vizitat în ordine de jos în sus)

if (n este un nod tabel de baza)

Cost (n) ← 0

Strat (n) ← subgraful care contine numai pe n

else

$\text{Cost}(m_i) \leftarrow \left(\sum_{j=1}^{k_i} Cost(w_{ij}) \right)$ + (costul aplicarii operatorului m_i la intrarile lui)

$\text{Strat}(m_i) \leftarrow \{m_i\} \cup \left(\bigcup_{j=1}^{k_i} \left(Strat(w_{ij}) \cup \{(w_{ij}, m_i)\} \right) \right)$

$Cost(n) \leftarrow \min_{1 \le j \le t} \{Cost(m_i)\}$

$Strat(n) \leftarrow Strat(m_i) \cup \{n\} \cup \{(m_i, n)\}$ unde **i** este valoarea pentru care s-a obtinut minimizarea

endif

endfor

Observam ca pentru fiecare $\mathbf{m_i}$, Cost (m_i) obtine costul celei mai bune strategii utili-zând $\mathbf{m_i}$ ca ultima garantie care calculeaza **n**, în timp ce Strat(m_i) realizeaza identificarea strategiei ca o multime de noduri si arce. De exemplu, algoritmul pentru un arbore Join la stânga este descris în figura 6:

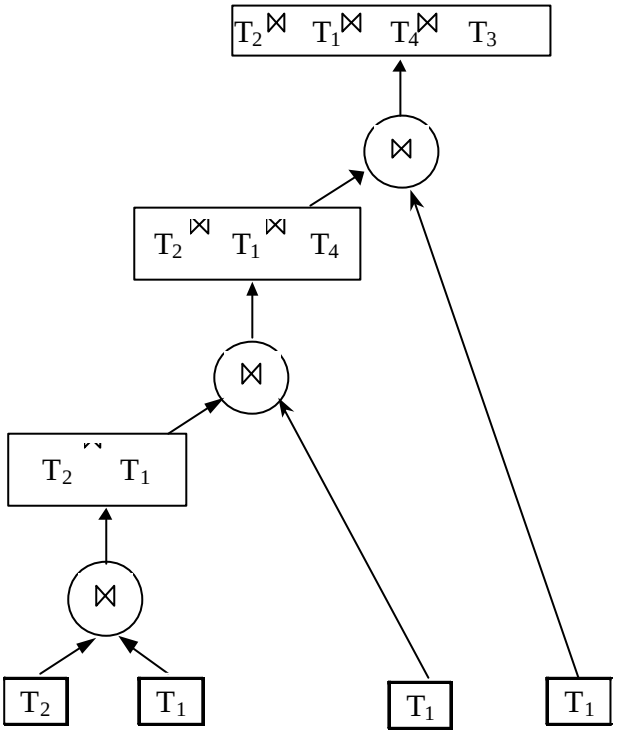

Figura 6

Pentru fiecare submultime nevida $Z \le \{T_2,...,T_k\}$ definim:
Cost(Z) = costul celei mai bune strategii care calculeaza ⋈
Strat(Z) = identifica strategia pentru care este calculat Cost(Z)

for i: = 1 to k
    Cost ($\{T_i\}$) $\leftarrow$ 0
    Strat ($\{T_i\}$) $\leftarrow T_i$
endfor
for s: = 2 to k
    for (fiecare s si submultime $Z \le (\{T_1,...,T_k\})$

$$Cost(Z) = \min_{w_j \in Z} \{ Cost(Z - \{w_j\}) + (\text{costul Join-ului rezultat a lui } \bowtie$$

$$(Z - \{w_j\}) \text{ cu } w_j)\}$$

$$Strat(Z) = Strat(Z - \{w_j\}) - w_j$$

endfor

endfor

Concluzii

Optimizarea presupune un efort considerabil din partea programatorului si necesita cu-noasterea temeinica a algebrei relationale, o proprietatilor operatorilor algebrici si a tehnicilor specifice de optimizare.