# CAPITOLUL 8

# OTIMIZAREA CERERILOR

Limbajul SQL este un limbaj de cereri de nivel inalt, neprocedural. Printr-o cerere se specifica **ce** doreste sa obtina utilizatorul si nu si **cum** se ajunge la rezultat. Modalitatea de obtinere a acestuia ramane in sarcina sistemului de gestiune care trebuie sa aleaga cea mai putin costisitoare cale. Costul este masurat de obicei prin timpul de executie care, asa cum spune si definitia bazei de date din capitolul introductiv, trebuie sa fie rezonabil.
Pentru reducerea timpului de raspuns sistemul efectueaza o serie de modificari ale cererii transformand-o intr-o cerere echivalenta dar care se poate executa mai rapid. Aceasta operatie poarta numele de **optimizarea cererii**.
Supuse optimizarii sunt in primul rand cererile de regasire de date (de tip SELECT in SQL). Trebuie avut insa in vedere ca si celelalte cereri de tip DML au nevoie de optimizari: pentru a actualiza sau sterge anumite linii dintr-o tabela ele trebuiesc intai localizate iar in cazul inserarilor trebuie verificat daca informatia nu exista deja in tabela.
In plus o serie de cereri de tip DDL si DCL includ operatii de regasire de date care implicit sunt optimizate de SGBD.

## 6.1. Elemente de cost si strategie

In cazul BDC, un aspect foarte important in contextul optimizarii cererilor este minimizarea numarului de accese la disc.
Sa luam exemplul unui produs cartezian intre doua relatii de cate doua atribute, fie ele R si S, avand **r** respectiv **s** tupluri. Fie **nr** respectiv **ns** numarul de tupluri din fiecare relatie care incap intr-un bloc pe disc, si **b** numarul de blocuri disponibile in memoria interna a calculatorului.
Sa presupunem ca procesul de calcul se desfasoara astfel: citim numarul maxim posibil de blocuri din R (adica b-1 blocuri) si pentru fiecare inregistrare din R citim in intregime, in ultimul bloc disponibil, relatia S.
Atunci:

Numarul de blocuri citite pentru a parcurge relatia R este
r/nr
Numarul de parcurgeri al relatiei S este
r/((b-1)*nr)
Pentru o parcurgere a relatiei S se citesc
s/ns blocuri

Deci numarul de accese la disc este in total:

Numarul de blocuri din R + Numarul de parcurgeri ale lui S * Numarul de blocuri din S adica:

(r/nr) + [ r/((b-1) * nr ] * (s/ns) care se poate rescrie si ca:

(r/nr) * (1 + s/((b-1) * ns))

In cazul relatiilor cu numar mare de tupluri aceasta expresie denota timpi mari pentru executia unei astfel de cereri. In cazul de mai sus, deoarece numarul de accese este mai puternic influentat de r/nr decat de s/ns, vom lua ca relatie R pe cea care are acest raport mai mic.

In implementarea joinurilor, se folosesc de asemenea diverse metode:

- sortarea uneia dintre relatii dupa atributele implicate in join (in cazul echijoinului)
- folosirea indecsilor (daca exista) pe campurile implicate in join, pentru acces direct la tuplurile care dau elemente ale rezultatului.
- micsorarea numarului de tupluri luate in calcul, prin aplicarea mai intai a selectiilor, daca acestea sunt prezente in cerere.

In general, exista o serie de principii (strategii) care duc la micsorarea numarului de accese la disc. Ele sunt urmatoarele (vezi si [Ul 82] - Ullman, J.D. Principles of Database Systems, Second Edition, Computer Science Press, 1982):

1. Realizarea selectiilor cit mai devreme posibil.
2. Combinarea anumitor selectii cu produse carteziene adiacente pentru a forma un join.
3. Combinarea secventelor de operatii unare (selectii si proiectii) intr-una singura (o selectie sau/si o proiectie)
4. Cautarea subexpresiilor comune, pentru a fi evaluate o singura data.
5. Folosirea indecsilor sau sortarea relatiilor, daca se obtine o crestere a performantelor.
6. Evaluarea diverselor strategii posibile inainte de a incepe procesul de calcul efectiv (in cazul in care sunt posibile mai multe metode de calcul) pentru a alege pe cea mai eficienta.

Pe baza acestor principii, exista o serie de algoritmi de optimizare pentru evaluarea expresiilor relationale.

## 6.2. Echivalente in algebra relationala

Pentru rescrierea unei expresii in algebra relationala in vederea optimizarii costurilor de evaluare se pot folosi urmatoarele echivalente:

Reguli de echivalenta a expresiilor:

***1. Comutativitatea joinului si a produsului cartezian:***

$$E1 \times E2 \equiv E2 \times E1$$
$$E1 \rhd\lhd E2 \equiv E2 \rhd\lhd E1$$
$$E1 \rhd\lhd_F E2 \equiv E2 \rhd\lhd_F E1$$

***2. Asociativitatea produsului cartezian si a joinului:***

$E1 \times (E2 \times E3) \equiv (E1 \times E2) \times E3$
$E1 \rhd\lhd (E2 \rhd\lhd E3) \equiv (E1 \rhd\lhd E2) \rhd\lhd E3$
$E1 \rhd\lhd_{F1} (E2 \rhd\lhd_{F2} E3) \equiv (E1 \rhd\lhd_{F1} E2) \rhd\lhd_{F2} E3$

***3. Cascada de proiectii:***

$\pi_{A1...An} (\pi_{B1...Bm}(E)) \equiv \pi_{A1...An} (E)$. Se presupune bineinteles ca $\{Ai\} \subseteq \{Bj\}$

***4. Cascada de selectii:***

$\sigma_{F1}(\sigma_{F2}(E)) \equiv \sigma_{F1 \wedge F2} (E)$

***5. Comutativitatea selectiilor cu proiectiile:***

Daca F contine doar atribute din A1...An atunci:
$\pi_{A1...An}(\sigma_F (E) ) \equiv \sigma_F (\pi_{A1...An}(E) )$

Sau, in cazul general – F contine si alte atribute, B1...Bm - atunci:
$\pi_{A1...An}(\sigma_F (E) ) \equiv \pi_{A1...An}(\sigma_F ( \pi_{A1...An, B1 ... Bm} (E) ) )$

***6. Comutativitatea selectiei cu produsul cartezian:***

Daca toate atributele din F fac parte din E1:
$\sigma_F (E1 \times E2) \equiv \sigma_F (E1) \times E2$

Daca $F = F1 \wedge F2$ cu F1 continand doar atribute din E1 si F2 doar din E2:
$\sigma_F (E1 \times E2) \equiv \sigma_{F1} (E1) \times \sigma_{F2} (E2)$

Daca $F = F1 \wedge F2$ cu F1 continand doar atribute din E1 dar F2 e o expresie generala:
$\sigma_F (E1 \times E2) \equiv \sigma_{F2} ( \sigma_{F1} (E1) \times E2)$

***7. Comutativitatea selectie – reuniune:***

$\sigma_F (E1 \cup E2) \equiv \sigma_F (E1) \cup \sigma_F (E2)$

***8. Comutativitatea selectie – diferenta:***

$\sigma_F (E1 - E2) \equiv \sigma_F (E1) - \sigma_F (E2)$

***9. Comutativitatea proiectie – produs cartezian:***

Daca in lista A1...An atributele B1...Bm sunt din E1 iar C1...Ck din E2 atunci:
$\pi_{A1...An}(E1 \times E2) \equiv \pi_{B1...Bm}(E1) \times \pi_{C1...Ck}(E2)$

***10. Comutativitatea proiectie – reuniune:***
$\pi_{A1...An}(E1 \cup E2) \equiv \pi_{A1...An} (E1) \cup \pi_{A1...An} (E2)$

## 6.4. Algoritm de optimizare a expresiilor relationale

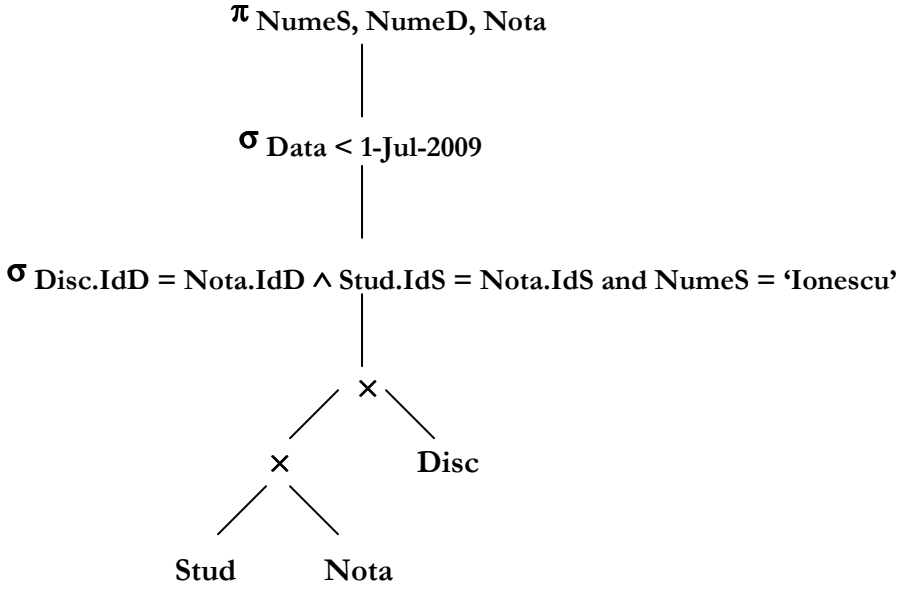
In lucrarea [Ul 82] este prezentat un algoritm de optimizare a expresiilor relationale. Acesta este urmatorul:

**Intrare:** un arbore reprezentand o expresie relationala.
**Iesire:** program pentru evaluarea expresiei.
**Metoda:**
1. Fiecare selectie este transformata folosind regula 4 intr-o cascada de selectii:

$$\sigma_{F1 \wedge F2 \wedge \ldots Fn}(E) \equiv \sigma_{F1}(\sigma_{F2}(\ldots(\sigma_{Fn}(E))\ldots))$$

2. Fiecare selectie este deplasata in jos folosind regulile 4-8 cât mai aproape de frunze.
3. Folosind regulile 3, 5, 9 si 10, fiecare proiectie este deplasata cât mai jos posibil in arborele sintactic. Regula 3 face ca unele proiectii sa dispara, regula 5 sparge o proiectie in doua astfel incat una poate migra spre frunze. Daca o proiectie ajunge sa fie dupa toate atributele ea dispare.
4. Cascadele de selectii si proiectii sunt combinate folosind regulile 3-5 intr-o singura selectie, o singura proiectie sau o selectie urmata de o proiectie. Aceasta abordare tine de eficienta: e mai putin costisitor sa se faca o singura selectie si apoi o singura proiectie decat daca s-ar alterna de mai multe ori selectii cu proiectii.
5. Nodurile interioare ale arborelui rezultat sunt impartite in grupuri. Fiecare nod intern care corespunde unei operatiuni binare devine un grup impreuna cu predecesorii sai imediati cu care sunt asociate operatiuni unare. Din bloc fac parte de asemenea orice lant de noduri succesoare asociate cu operatiuni unare si terminate cu o frunza cu exceptia cazului când operatia binara este un produs cartezian neurmat de o selectie cu care sa formeze un echijoin.
6. Se evalueaza fiecare grup astfel incat niciunul nu este evaluat inaintea descendentilor sai. Rezulta astfel un program de evaluare a expresiei relationale initiale.

Exemplu: Fie o baza de date care implementeaza o diagrama entitate asociere formata din 2 entitati (Student, Disciplina) si o asociere binara multi-multi Nota continand notele studentilor si data obtinerii lor ca atribute proprii.

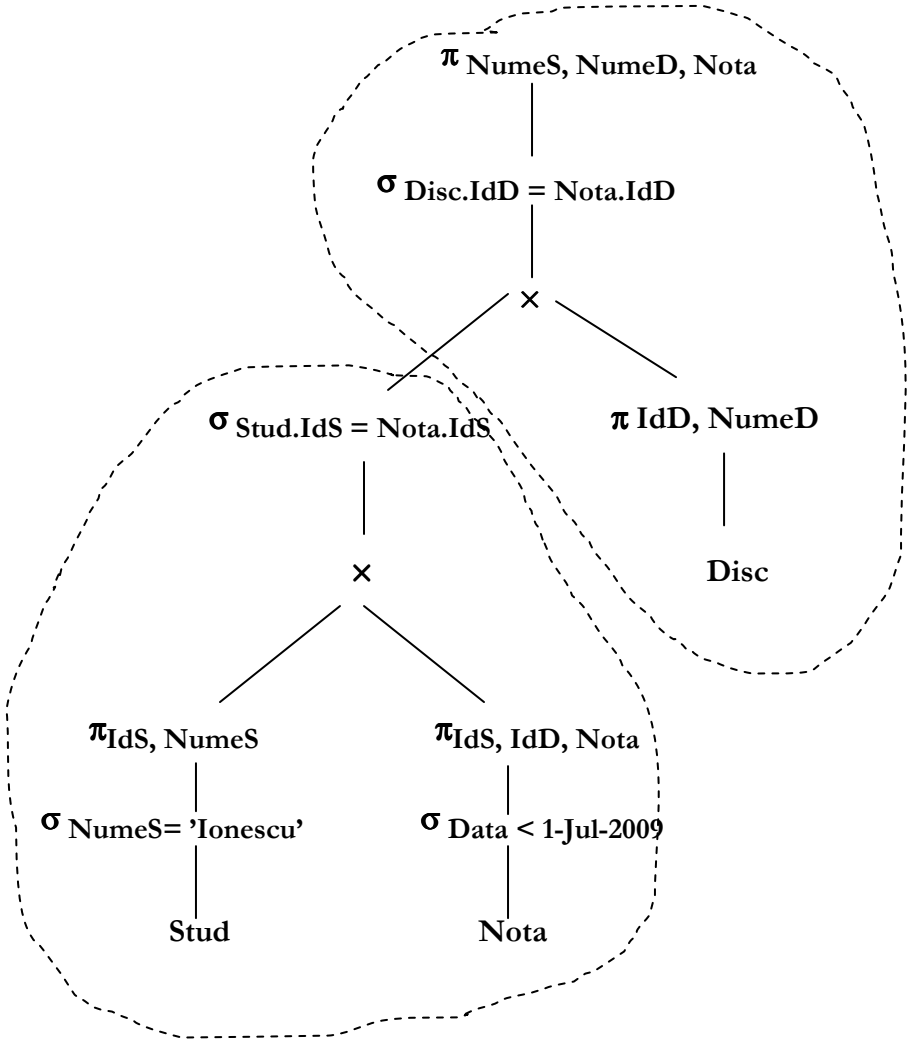


Expresia relationala de mai sus afiseaza numele studentului si notele sale pentru studentul cu numele 'Ionescu' si note obtinute obtinute pana la 1 iulie 2009.
Structura simplificata a bazei de date este urmatoarea:

Stud (IdS, NumeS)
Disc (IdD, NumeD)
Nota (IdS, IdD, Nota, Data)

Aplicand algoritmul de mai sus:

- Selectia dupa data va ajunge pe ramura tabelei Disc
- Conditia de join se va sparge in trei, doua conditii de join separate, una pentru echijoinul Stud cu Nota si alta pentru echijoinul rezultatului primului join cu Disc si o a treia dupa numele studentului care coboara pe ramura respectiva.
- Regula 5 generalizata va duce la adaugarea unor proiectii care lasa sa treaca mai departe de la frunze doar atributele necesare operatiilor ulterioare.
- Rezultatul obtinut este urmatorul:




Se observa formarea a doua grupuri care se evalueaza in ordinea mentionata in algoritm: intai grupul de jos si apoi grupul de sus care foloseste rezultatul primului grup. Pentru fiecare grup produsul cartezian si selectia se pot combina intr-un echijoin.

## 6.4. Studiu de caz: System R

Sistemul de gestiune a bazelor de date System R a fost dezvoltat intre 1975 si 1979 in laboratoarele firmei IBM de la San Jose, California. Din el au derivat urmatoarele produse comerciale:

- SQL/DS aparut in 1981 pentru sistemul de operate DOS/VSE si in 1983 pentru VM/CMS.
- DB2 aparut in 1983 pentru sistemul de operare MVS (produs comercializat si azi de IBM in versiunile sale ulterioare)
- Trebuie mentionat ca din acest sistem s-a inspirat initial si sistemul de gestiune Oracle dezvoltat independent de actuala Oracle Corp.

Limbajul de cereri al acestui sistem este SQL (limbaj dezvoltat tot de IBM).

**Structura stocarii datelor**

Datele sunt stocate intr-o multime de adrese logice numite segmente. Un segment contine mai multe relatii (initial s-a dorit stocarea mixata a relatiilor pentru a accelera operatia de join dar apoi s-a renuntat).
Exista mai multe tipuri de segmente: pentru relatii, pentru relatii temporare, etc.
Paginile adiacente ale unei relatii sunt stocate in segment fizic adiacente.
Pentru fiecare segment exista o tabela de traducere de la adresa logica la adresa de disc. Aceasta tabela este divizata in blocuri de lungime fixa.
Exista doua zone tampon: una pentru blocurile tabelei de pagini si alta pentru paginile propriu-zise.

Paginile si blocurile ramin in memorie pina la eliberarea lor explicita. Eliberarea unei pagini o face candidata la inlocuire. Inlocuirea vizeaza pagina libera cel mai putin recent ceruta.

**Cai de acces**

In System R exista doua feluri de indecsi:

- indecsi pe un atribut (ascendent/descendent)
- indecsi pe o combinatie de atribute

Indecsii permita accesul direct la:

- unul sau mai multe tuple cu aceeasi valoare a cheii
- o multime de tuple, avand chei intr-o plaja data
- parcurgerea secvential-sortata a unei relatii (in ordinea indexului)

Accesul se face pe baza unui predicat de cautare rezultat dintr-o cerere care cuprinde cel putin un cimp indexat.
Indecsii sunt multinivel, sub forma de arbori-B de pagini. Fiecare tuplu are o adresa logica unica numita TID (tuple identifier = identificator de tuplu) care este format din numarul de pagina si adresa indirecta in pagina.
Nodurile frunza ale arborelui B - index contin o valoare de index si o lista de TID-uri. Paginile frunza sunt legate sunt legate intr-o lista dubla pentru a se putea efectua parcurgerea secventiala in ordinea indexului.
In cazul cautarilor multicriteriale, se foloseste un mecanism de codificare pentru a genera din mai multe cimpuri o cheie unica.

In cazul System R indecsii pot fi de doua tipuri:

1. index grupant, in care caz tuplele relatiei sunt plasate fizic in ordinea specificata de index. In acest mod, tuplele cu valori apropiate ale cheii de indexare sunt apropiate si fizic (eventual in aceeasi pagina). In acest fel este favorizata parcurgerea secventiala a tuplelor relatiei in ordinea data de index.
2. index non-grupant, in care caz numarul de accese la memoria externa in cazul parcurgerii secventiale in ordinea indexului este mai mare.

Cautarile in cazul System R pot fi:

1. Fara folosirea indexului (cautare in segment)
2. Cu folosirea indexului (cautare prin index)

**Arhitectura System R**

SISTEMUL RELATIONAL DE DATE (SRD)
- analiza
- autorizare
- constringeri de integritate
- meta - baza de date
- selectia cailor de acces

SISTEMUL DE STOCARE SI REGASIRE (SRR)
- gestiunea memoriei
- gestiune indecsi
- controlul concurentei
- jurnalizare si restart

Sistemul de stocare si regasire (SSR) este un SGBD de nivel scazut. Are o interfata interna continand operatori de acces "un tuplu la un moment dat" la relatie.
Apelul la acesti operatori se face specificand segmentul si indexul utilizat. Interfata permite parcurgerea tuplelor conform unei cai de acces specificate.
Functiile asigurate de SSR sunt:

- alocarea memoriei secundare
- alocarea memoriei tampon (buffere)
- controlul tranzactiilor (concurenta, restart)
- mentinerea automata a indecsilor

Sistemul relational de date (SRD) ofera interfata externa SQL care poate fi apelata dintr-o tranzactie sau dintr-un limbaj de programare.
Functiile asigurate sunt:

- compilarea comenzilor SQL rezultand module de acces (limbaj SSR).
- controlul executiei programelor care contin cereri SQL si a cererilor utilizatorilor care lucreaza interactiv.
- Realizeaza functii de definitie date, control, manipulare date, optimizare cereri, etc.

Abordarea este compilativa si nu interpretativa. O comanda SQL este compilata in 3 etape:

- analiza sintactica
- optimizarea cererii
- generarea modulului de acces

**Optimizari in System R**

In [Ul 82] este descris un algoritm de oprimizare a cererilor folosit de System R. El rezolva problema in cazul cererilor simple, de forma:

```
SELECT A1, A2, ..., An
FROM R
WHERE P1 AND P2 AND ...
```

unde Ai sunt atribute ale relatiei R iar Pk sunt predicate.

Algoritmul este urmatorul:
**Intrare:**

- cerere SQL ca mai sus
- informatii despre indecsii care exista pe relatia R.
- pentru fiecare index, valoarea estimata a dimensiunii imaginii sale. Dimensiunea imaginii unui index dupa atributul A este egala cu numarul de valori distincte memorate pentru acel atribut.
- valoarea lui T = numarul de tuple din R.
- valoarea lui B = numarul de blocuri teoretic necesar pentru memorarea relatiei R.

**Iesire:** O metoda de calcul al respunsului pentru cererea SQL.

**Metoda:** Se foloseste una dintre metodele de mai jos pentru obtinerea fie a lui R, fie a lui R asupra caruia s-a aplicat o selectie dupa unul dintre predicatele din cerere. Pentru acele metode care implica alegerea indexului sau predicatului de folosit, se considera toate posibilitatile de alegere. Metodele sunt listate in ordinea in care trebuiesc parcurse, dar in toate trebuie sa se foloseasca parametrii T si B precum si dimensiunea imaginii indecsilor pentru estimarea costului. Metoda cu cel mai scazut cost estimat va fi rezultatul algoritmului.
***Metoda 1.*** Daca unul dintre predicate este de forma A = c si pentru A exista un index grupant, se aplica acest predicat si dupa aceea se aplica celelalte predicate tuplelor obtinute.
Daca I este dimensiunea imaginii indexului mentionat, se citesc aproximativ T/I tuple in medie. Cum indexul este grupant, rezulta ca se citesc in medie (B/I) blocuri ale relatiei R.
***Metoda 2.*** Daca unul dintre predicate este de forma A op c, unde op este un operator de comparatie (<, <=, >, >=) si pentru A exista un index grupant, se aplica acest predicat.
Costul acestei metode este este de B/2, deoarece in medie citim T/2 tuple si cum indexul este grupant, acestea se afla memorate in B/2 blocuri.
***Metoda 3.*** Daca unul dintre predicate este de forma A = c si pentru A exista un index non-grupant, se aplica acest predicat si dupa aceea se aplica celelalte predicate tuplelor obtinute. Daca I este dimensiunea imaginii indexului mentionat, se citesc aproximativ T/I tuple in medie. Cum indexul este non-grupant, tuplele se pot afla in diverse blocuri, deci costul acestei metode este de T/I.

***Metoda 4.*** Daca relatia R este memorata singura in fisier (nu intretesut cu alte relatii), se citesc toate tuplele lui R si se aplica predicatele fiecarui tuplu. Costul acestei metode este B, deoarce se citesc toate blocurile acelei relatii.
***Metoda 5.*** Daca R nu este memorata singura in fisier, dar exista un index grupant dupa unul dintre atribute, fie ca sunt cele din conditia de selectie sau nu, se foloseste acel index pentru a obtine toate tuplele lui R si li se aplica predicatele. Costul metodei este de asemenea B, deoarece un index grupant garanteaza ca se pot obtine tuplele unei relatii in tot atitea accese cite sunt necesare pentru memorarea sa.
***Metoda 6.*** Daca A op c este un predicat unde op este un operator de comparatie (<, <=, >, >=) si pentru A exista un index non-grupant, se foloseste acel index pentru a gasi toate tuplele lui R care satisfac atributul si se aplica apoi celelalte predicate. Costul este T/2, deoarece indexul fiind non-grupant, tuplele se pot gasi in blocuri diferite si in medie trebuiesc citite jumatate dintre ele.
***Metoda 7.*** Se foloseste un index non-grupant pentru a gasi toate tuplele lui R si li se aplica predicatele. Costul metodei este T.
***Metoda 8.*** Daca nici una dintre metodele de mai sus nu poate fi folosita, se parcurg blocurile care pot contine tuple din R si se gasesc acele tuple. Costul acestei metode este mai mare sau egal cu T.