Evaluarea si optimizarea interogarilor.
Proprietatile operatorilor algebrei relationale.
Reguli de optimizare.
Arbori algebrici. Operatori arbori algebrici.
Exemplu.

Evaluarea si optimizarea interogarilor

In momentul in care se implementeaza o cerere (interogare) in SQL, sistemul de gestiune (Oracle SQL)

- **ANALIZEAZA** semantic (programul ruleaza si compileaza fara erori) si sintactic (programul ruleaza si compileaza fara erori, dar este corect si din punct de vedere sintactic) pentru a verifica corectitudinea cererii. In acest moment se verifica si accesul la datele implicate in cererea SQL;

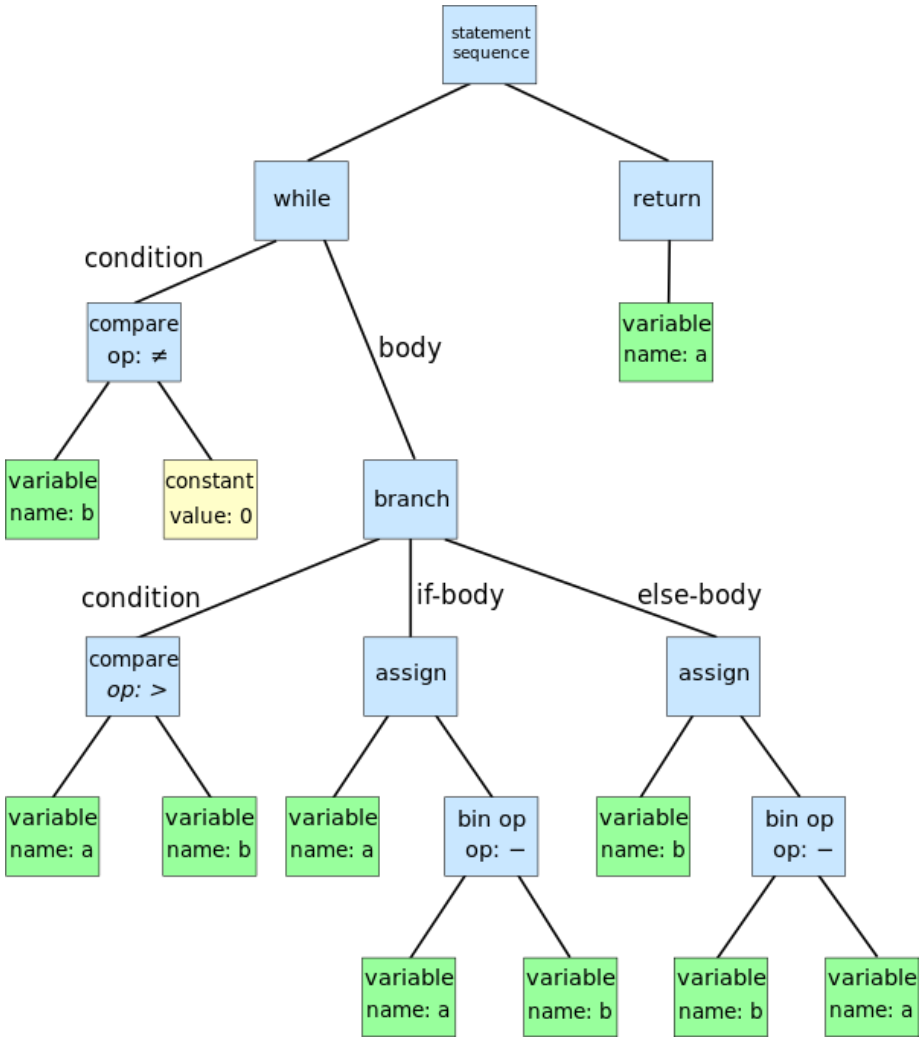
- Dupa etapa de analizare urmeaza etapa de **ORDONARE** in care cererea este descompusa intr-o multime de operatii apartinand algebrei relationale (SELECT, PROJECT, JOIN, DIVISION, UNION, INTERSECT, PRODUCT, DIFFERENCE), dupa care se stabileste o ordine de executie optima a acestor operatii. In acest moment se obtine un **plan de executie al cererii**;

- In final, urmeaza **EXECUTIA** cererii si obtinerea rezultatului final;

In general, compilatorul efectueaza trei tipuri de analiza: **semantica, sintactica, lexicala.**

Se verifica daca programul ruleaza si compileaza fara erori. Erorile semantice sunt erorile de compilare care nu sunt erori de sintaxa (ex: nu avem tipuri de date compatibile, nu se realizeaza o conversie automata, nu are acces la o proprietate, etc). In final, pentru realizarea tuturor verificarilor, compilatorul realizeaza un arbore, dupa cum se observa in exemplul urmator:

```
while b ≠ 0
  if a > b
    a = a − b
  else
    b = b − a
return a
```



In acelasi mod executa o cerere si compitalorul SQL, convertind cererea utilizatorului intr-o cerere optima cu ajutorul unui arbore algebric.

cerere -> arbore algebric -> plan de executie -> optimizare

Cu ajutorul **planului de executie** se realizeaza o evaluare a cererii in vederea realizarii unor optimizari. Arborele algebric nefiind unic, pot exista planuri de executie diferite, dar care in final sa produca acelasi rezultat. Scopul optimizarii este acela de a identifica planul de executie optim (de cost minim).

Astfel, o **cerere SQL** se poate scrie sub forma unei **expresii a algebrei relationale**, formata din relatii si operatii specifice algebrei relationale, dupa care se poate reprezenta grafic sub forma de arbore, numit **arbore algebric**, in care nodurile corespund operatorilor algebrei relationale utilizati in cadrul cererii, urmand etapa de **evaluare** si **optimizare** a cererii.

Optimizarea cererilor utilizand algebra relationala se realizeaza pornind de la scrierea cererii folosind expresii algebrice, dupa care se aplica transformari echivalente, dar mult mai optime.

Pentru optimizare se utilizeaza si **proprietatile operatorilor algebrei relationale**.

Proprietatile operatorilor algebrei relationale

Proprietatea 1. Comutativitatea operaţiilor de *join* şi produs cartezian:

JOIN($R1$, $R2$) = JOIN($R2$, $R1$)

$R1 \times R2 = R2 \times R1$

Proprietatea 2. Asociativitatea operaţiilor de *join* şi produs cartezian:

JOIN(JOIN($R1$, $R2$), $R3$) = JOIN($R1$, JOIN($R2$, $R3$))

$(R1 \times R2) \times R3 = R1 \times (R2 \times R3)$

Proprietatea 3. Compunerea proiecţiilor:

$\Pi_{A1,...,Am}(\Pi_{B1,...,Bn}(\boldsymbol{R})) = \Pi_{A1,...,Am}(\boldsymbol{R})$,
unde $\{A_1, A_2,...,A_m\} \subseteq \{B_1, B_2,...,B_n\}$.

SAU:

PROJECT(PROJECT(**R**, B_1, B_2,...,B_n), A_1, A_2,...,A_m)

Rezulta o relatie din care se vor prelua atributele A_1, A_2,...,A_m`

SAU:

R1 = PROJECT(**R**, B_1, B_2,...,B_n)
R2 = PROJECT(R1, A_1, A_2,...,A_m)

EXEMPLU:

B_1, B_2,...,B_n

R1 = PROJECT(EMPLOYEES, last_name, job_id, department_id)
R2 = PROJECT(R1, last_name, job_id)

A_1, A_2,...,A_m

Se observa si proprietatea $\boldsymbol{\{A_1, A_2,...,A_m\} \subseteq \{B_1, B_2,...,B_n\}}$, adica A este subset al lui B (last_name si job_id este o submultime a multimii -> last_name job_id, department_id).

Proprietatea 4. Compunerea selecţiilor:

$\sigma_{cond1}(\sigma_{cond2}(R)) = \sigma_{cond1 \wedge cond2}(R) = \sigma_{cond2}(\sigma_{cond1}(R))$,
unde am notat prin *cond* condiţia după care se face selecţia.

SAU:

SELECT(SELECT(R, cond2), cond1) =
SELECT(R, cond1 and cond2) = SELECT(SELECT(R, cond1), cond2)

EXEMPLU:

Sa se afiseze angajatii care au salariul > 5000 si codul jobului ‘SA_REP’.

R1 = SELECT**(**SELECT**(**EMPLOYEES, salary > 5000),
job_id = ‘SA_REP’**))**

⇔ R1 = SELECT(EMPLOYEES, salary > 5000 AND job_id = ‘SA_REP’)

⇔ R1 = SELECT(SELECT**(**EMPLOYEES, job_id = ‘SA_REP’**)** ,
salary > 5000**)**

Proprietatea 5. Comutarea selecţiei cu proiecţia:

$\Pi_{A1,...,Am}(\sigma_{cond}(R)) = \sigma_{cond}(\Pi_{A1,...,Am}(R))$,

unde condiţia după care se face selecţia implică numai atributele $A_1,...,A_m$.

Dacă condiţia implică şi atributele $B_1,...,B_n$, care nu aparţin mulţimii $\{A_1,...,A_m\}$, atunci:

$\Pi_{A1,...,Am}(\sigma_{cond}(R)) = \Pi_{A1,...,Am}(\sigma_{cond}(\Pi_{A1,...,Am,B1,...,Bn}(R)))$

SAU:
PROJECT(SELECT(R, cond), $A_1,...,A_m$) =

SELECT(PROJECT(R, $A_1,...,A_m$), cond)

EXEMPLU:

R1 = SELECT(EMPLOYEES, salary > 5000)
R2 = PROJECT(R1, last_name, salary)

SAU:

R1 = PROJECT(EMPLOYEES, last_name, salary)
R2 = SELECT(R1, salary > 5000)

Proprietatea 6. Comutarea selecţiei cu produsul cartezian:

Dacă toate atributele menţionate în condiţia după care se face selecţia sunt atribute ale relaţiei R1, atunci:

$\sigma_{cond}\ (R1 \times R2) = \sigma_{cond}\ (R1) \times R2$

SAU:

SELECT(PRODUCT(R1,R2), cond) =

PRODUCT(SELECT(R1, cond), R2)

Dacă condiţia este de forma cond1∧cond2 şi dacă cond1 implică numai atribute din R1, iar cond2 implică numai atribute din R2, atunci:

$\sigma_{cond}\ (R1 \times R2) = \sigma_{cond1}\ (R1) \times \sigma_{cond2}\ (R2)$

SAU:

SELECT(PRODUCT(R1,R2), cond) =

PRODUCT**(** SELECT(R1, cond1), SELECT(R2, cond2) **)**

Dacă cond1 implică numai atribute din R1, iar cond2 implică atribute atât din R1 cât şi din R2, atunci:

$\sigma_{cond}(R1 \times R2) = \sigma_{cond2}(\sigma_{cond1}(R1) \times R2)$

SAU:

SELECT(PRODUCT(R1,R2), cond) =

SELECT**(** PRODUCT **(** SELECT **(**R1, cond1**)**, R2**)** , cond2**)**

SAU:

R = SELECT(R1, cond1)
S = PRODUCT(R, R2)
Rezultat = SELECT(S, cond2)

Proprietatea 7. Comutarea selecţiei cu reuniunea:

$\sigma_{cond}(R1 \cup R2) = \sigma_{cond}(R1) \cup \sigma_{cond}(R2)$

SAU:

SELECT(UNION(R1,R2), cond) =

UNION(SELECT(R1, cond), SELECT(R2, cond))

SAU:

R = SELECT(R1, cond)
S = SELECT(R2, cond)
Rez = UNION(R1,R2)

Proprietatea 8. Comutarea selecţiei cu diferenţa:

$\sigma_{cond}(R1 - R2) = \sigma_{cond}(R1) - \sigma_{cond}(R2)$

SAU:

SELECT(DIFFERENCE(R1,R2), cond) =

DIFFERENCE(SELECT(R1, cond), SELECT(R2, cond))

SAU:

R = SELECT(R1,cond)
S = SELECT(R2, cond)
Rez = DIFFERENCE(R,S)

Proprietatea 9. Comutarea proiecţiei cu produsul cartezian:

Dacă $A_1,...,A_m$ este o listă de atribute ce apar în schemele relaţionale $R1$ şi $R2$ şi dacă lista este formată din atribute aparţinând lui $R1$ (notate prin $B_1,...,B_n$) şi din atribute aparţinând lui $R2$ (notate prin $C_1,...,C_k$) atunci:

$\Pi_{A1,...,Am}(R1 \times R2) = \Pi_{B1,...,Bn}(R1) \times \Pi_{C1,...,Ck}(R2)$

SAU:

PROJECT(PRODUCT(R1,R2), $A_1,...,A_m$) =

PRODUCT(PROJECT(R1, $B_1,...,B_n$),
PROJECT(R2, $C_1,...,C_k$)
)

Proprietatea 10. Comutarea proiecţiei cu reuniunea:

$\Pi_{A1,...,Am}(R1 \cup R2) = \Pi_{A1,...,Am}(R1) \cup \Pi_{A1,...,Am}(R2)$

SAU:

PROJECT(UNION(R1,R2), A_1,...,A_m) =

UNION(PROJECT(R1, A_1,...,A_m)
PROJECT(R2, A_1,...,A_m)
)

Proprietatea 11. Compunerea proiecţiei cu operaţia *join*:

Dacă A_1,...,A_m este o listă de atribute ce apar în schemele relaţionale $R1$ şi $R2$ şi dacă lista este formată din atribute aparţinând lui $R1$ (notate prin B_1,...,B_n) şi din atribute aparţinând lui $R2$ (notate prin C_1,...,C_k) atunci:

$\Pi_{A1,...,Am}(\text{JOIN}(R1,R2,D)) = \Pi_{A1,...,Am}(\text{JOIN}(\Pi_{D,B1,...,Bn}(R1), \Pi_{D,C1,...,Ck}(R2),D),$

unde am notat prin JOIN($R1$, $R2$, D) operaţia de compunere naturală între $R1$ şi $R2$ după atributul comun D.

SAU:

A_1,...,A_m atribute prezente in R1 si R2

B_1,...,B_n atribute prezente in R1

C_1,...,C_k atribute prezente in R2

PRODUCT(JOIN(R1,R2,D), A_1,...,A_m) =

PROJECT(JOIN(PROJECT(R1, B_1,...,B_n),
PROJECT(R2, C_1,...,C_k),
D),
A_1,...,A_m)

SAU:

R = PROJECT(R1, D, B_1,...,B)
S = PROJECT(R2, D, C_1,...,C_k)
T = JOIN(R, S, D)
Rez = PROJECT(T, A_1,...,A_m)

Proprietatea 12. Compunerea selecţiei cu operaţia *join*:

$$\sigma_{cond}(JOIN(R1, R2, D)) = \sigma_{cond}(JOIN(\Pi_{D,A}(R1), \Pi_{D,A}(R2), D)),$$

unde *A* reprezintă atributele care apar în condiţia după care se face selecţia.

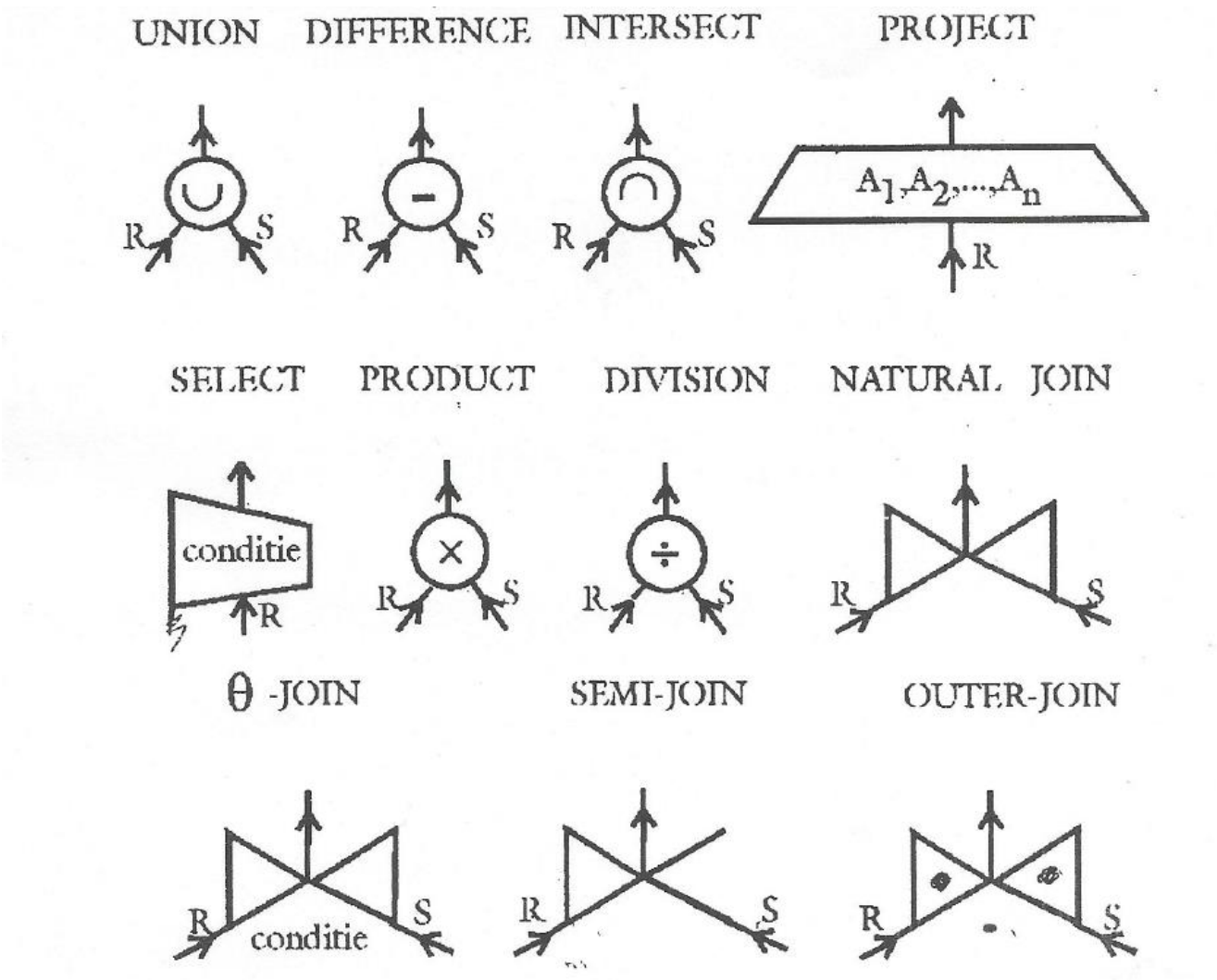
SAU:

```
SELECT( JOIN ( PROJECT(R1, D, A),
               PROJECT(R2, D, A),
               D
             ) ,
        cond)
```

Reguli de optimizare

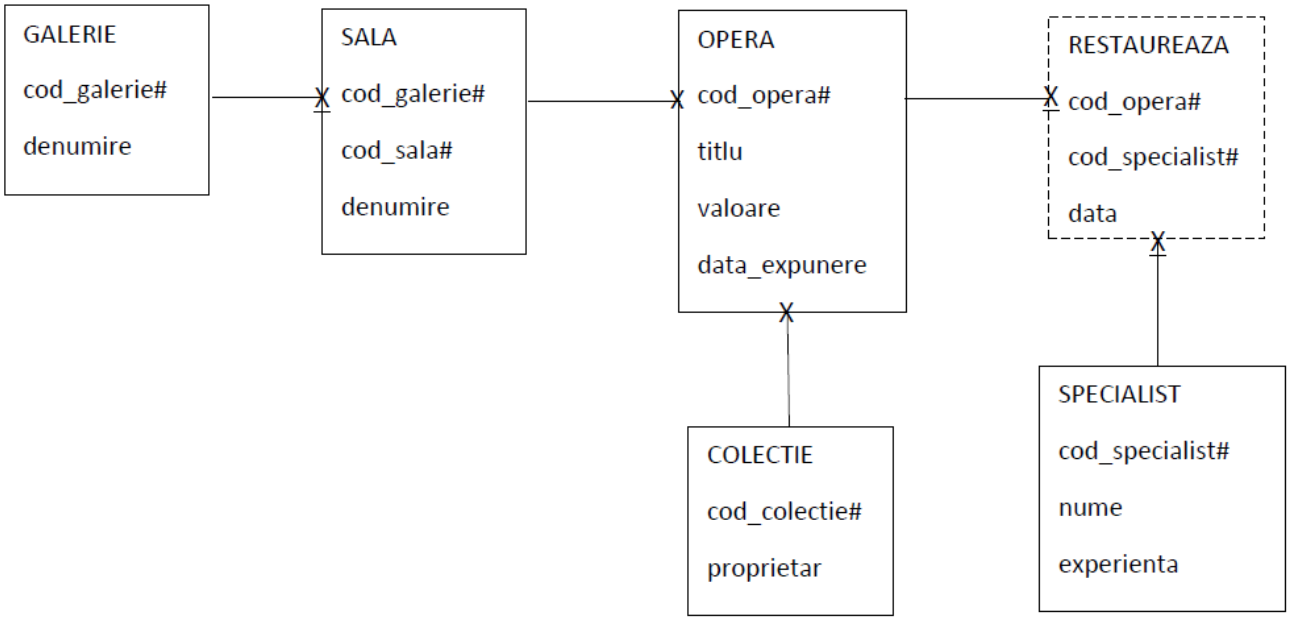
- **Regula de optimizare 1.** Selecţiile se execută cât mai devreme posibil.
- **Regula de optimizare 2.** Produsele carteziene se înlocuiesc cu *join*-uri, ori de câte ori este posibil
- **Regula de optimizare 3.** Dacă sunt mai multe *join*-uri atunci cel care se execută primul este cel mai restrictiv.
- **Regula de optimizare 4.** Proiecţiile se execută la început pentru a îndepărta atributele nefolositoare.

Arbori algebrici. Operatori arbori algebrici



Exemplu

Sa se obtina titlul si valoarea operelor de arta expuse in galeria avand codul G1 care fac parte din colectiile ce apartin proprietarului cu numele King si care au fost restaurate dupa data 15/06/2000.



Cerere SQL :

```
SELECT titlu, valoare
FROM opera o JOIN colectie c ON (o.cod_colectie = c.cod_colectie)
             JOIN restaureaza r ON (o.cod_opera = r.cod_opera)
WHERE o.cod_galerie = 'G1'
  AND c.proprietar = 'King'
  AND r.data > to_date('05/06/2000', 'dd/mm/yyyy');
```

Expresie algebrica*:*

R1 = SELECT(OPERA, cod_galerie = ‘G1’)
R2 = PROJECT(R1, cod_opera, titlu, valoare, cod_colectie)
R3 = SELECT(COLECTIE, proprietar = ‘King’)
R4 = PROJECT(R3, cod_colectie)
R5 = SEMIJOIN(R2, R4, cod_colectie)
R6 = SELECT(RESTAUREAZA, data > 05/06/2000)
R7 = PROJECT(R6, cod_opera)
R8 = SEMIJOIN(R5, R7, cod_opera)
Rezultat = R9 = PROJECT(R8, titlu, valoare)

Arbore algebric: