BAZE DE DATE

CURS 7

Evaluarea şi optimizarea interogărilor

Procesarea interogărilor

- O **expresie a algebrei relaţionale** este constituită din **relaţii** legate prin **operaţii** din algebra relaţională;
- O expresie se poate reprezenta **grafic** cu ajutorul unui **arbore**, numit **arbore algebric**, în care nodurile corespund operatorilor din cadrul expresiei respective;
- Evaluarea unei expresii presupune efectuarea prelucrărilor indicate de operatorii din expresie în ordinea apariţiilor sau în ordinea fixată prin paranteze;
- **Rezultatul** evaluării unei expresii este o **relaţie derivată** din relaţiile menţionate ca operanzi în cadrul expresiei;
- Două expresii sunt echivalente, dacă în urma evaluării lor se obţine ca rezultat aceeaşi relaţie;

Procesarea interogărilor

- În majoritatea sistemelor de gestiune, în special în cele relaţionale, interfaţa cu utilizatorul este de tip **neprocedural**;
  - Utilizatorul defineşte datele pe care doreşte să le vizualizeze fără a da algoritmii de acces (**CE**, nu **CUM**)
  - Sistemul trebuie să convertească cererea utilizatorului:
    - într-o cerere optimală;
    - în proceduri de acces optimal la datele fizice;
- Garantarea absolută a performanţelor optime pentru procesorul limbajului relaţional este imposibilă;
- Corectă ar fi utilizarea cuvântului „**ameliorare**“ în locul cuvântului „**optimizare**“;

Procesarea interogărilor

În momentul în care se implementează o cerere (interogare) în SQL, sistemul de gestiune (Oracle Database) evaluează cererea.

Evaluarea unei interogări se efectuează în trei etape:

- **Analiza cererii** – presupune **studierea semantică şi sintactică** a cererii pentru a verifica corectitudinea sa şi a simplifica criteriul de căutare.

Semantic – se verifică dacă programul rulează și compilează fără erori. **Sintactic** – se verifică dacă programul rulează și compilează fără erori, dar și dacă este corectă sintaxa utilizată (pentru a verifica corectitudinea cererii). În acest moment se verifică și accesul la datele implicate în cererea SQL.

Procesarea interogărilor

- **Ordonanţarea/Ordonarea** – presupune **descompunerea cererii într-o mulţime de operaţii elementare** şi determinarea unei ordini optimale a acestor operaţii.

Operaţiile sunt, în general, cele ale **algebrei relaţionale** (SELECT, PROJECT, JOIN, DIVISION, UNION, INTERSECT, PRODUCT, DIFFERENCE), după care se stabilește o ordine de execuție optimă a acestor operații.

La sfârşitul etapei se obţine un **plan de execuţie al cererii**.

Procesarea interogărilor

- **Execuţia** – presupune **efectuarea** (paralelă şi/sau secvenţială) operaţiilor elementare furnizate de planul de execuţie pentru a obţine **rezultatul cererii**.

Procesarea interogărilor

- Presupunem că utilizatorul transmite sistemului de gestiune o **cerere** exprimată prin **comenzi SQL**
- Pentru a răspunde cererii, SGBD-ul trebuie să înţeleagă cererea utilizatorului
- Cererea trebuie să:
  - fie corectă sintactic;
  - aibă datele **disponibile** utilizatorului;
  - localizeze datele analizând diferite **drumuri de acces** la ele;

Procesarea interogărilor

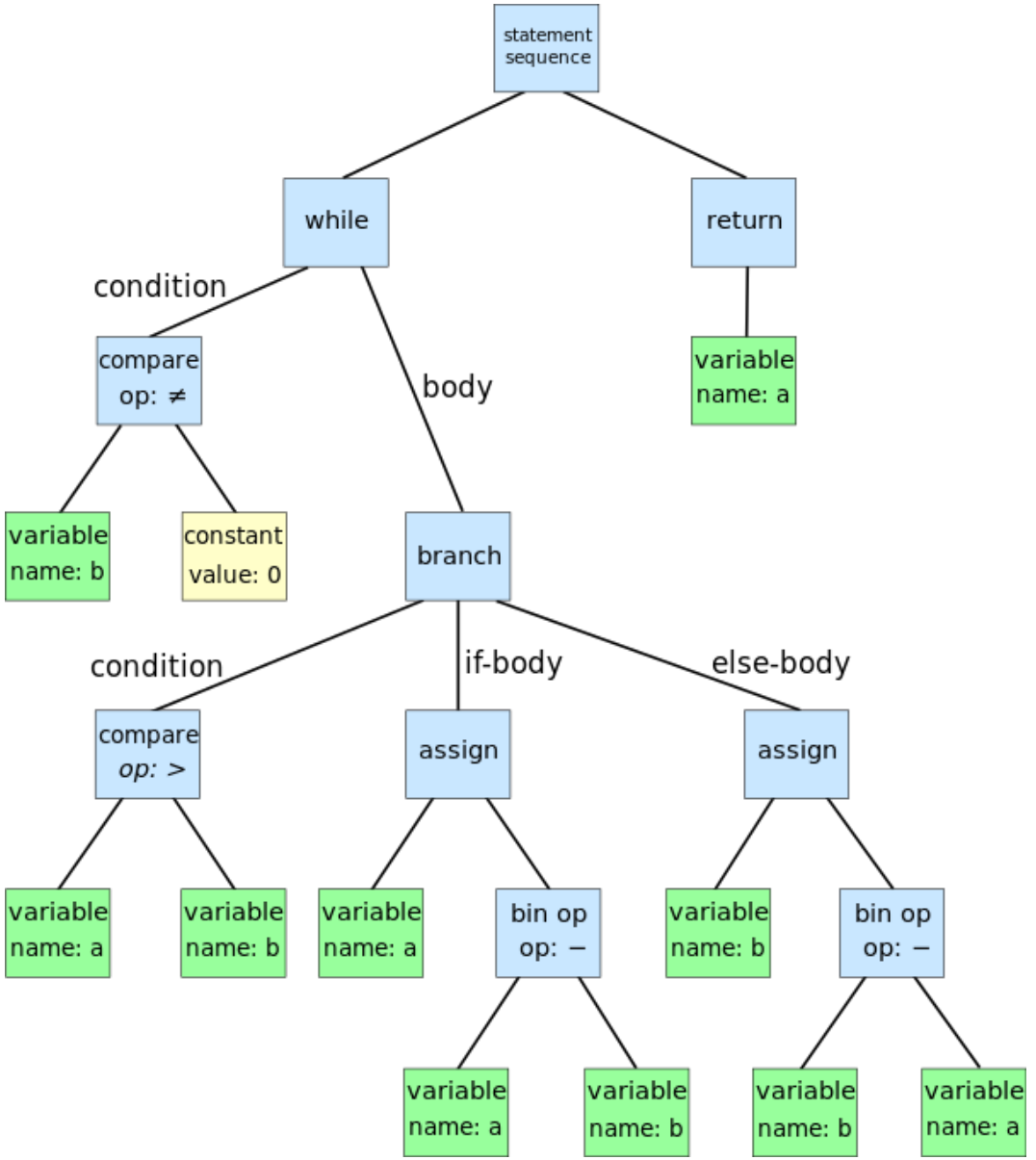
- Aceste funcţii sunt realizate de SGBD cu ajutorul a **două module funcţionale** care comunică permanent:
  - **analizorul cererilor**, care asigură:
    - verificarea sintactică şi semantică a cererii;
    - localizarea datelor implicate în cerere (găsirea adresei blocurilor ce conţin datele);
    - furnizarea planului de execuţie;
  - **administratorul datelor** (executorul), care execută efectiv cererea (primeşte planurile de execuţie furnizate de modulul de optimizare şi le execută)
    - Execuţia presupune căutarea blocurilor ce conţin datele şi transferul blocurilor în memoria *cache;*

Procesarea interogărilor

- 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:

Procesarea interogărilor

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



Procesarea interogărilor

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

Ideea generală:

cerere → arbore algebric (nu este unic) → plan de execuție → optimizare

Procesarea interogărilor

- 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);
- Un plan de execuţie implică:
  - **o secvenţă de paşi pentru evaluarea cererii** - în mod obişnuit, fiecare pas din planul de execuţie corespunde unei operaţii relaţionale;
  - **metoda care va fi folosită pentru evaluarea operaţiei** - de obicei, pentru o operaţie relaţională dată, există mai multe metode ce pot fi folosite pentru evaluarea acesteia;

Procesarea interogărilor

- Două planuri de execuţie diferite care au întotdeauna acelaşi rezultat se numesc **echivalente**. Planuri de execuţie echivalente pot avea diferite costuri;
- Scopul optimizării cererilor este de a găsi, printre diversele planuri de execuţie echivalente, pe acela de cost minim;
- Într-un sistem centralizat, costul evaluării unei cereri este suma a două componente:
  - **costul I/O** (transferuri de date) + **costul CPU** (verificare de condiţii, operaţii *join,* etc.)

Ordinea de execuţie a operaţiilor

- O interogare constă dintr-un număr de operaţii
  - ordinea în care se efectuează operaţiile are un rol important în evaluarea costului necesar realizării interogării;
- Există **două modalităţi** de abordare pentru a determina ordinea de execuţie a operaţiilor:
  - algebric;
  - bazat pe estimarea costului;
- Ambele folosesc o mulţime de **reguli** care permit **transformarea unui plan de execuţie** (reprezentat ca o expresie scrisă în termenii algebrei relaţionale) în altul, echivalent

Ordinea de execuţie a operaţiilor

- 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**;

Proprietăţile operatorilor algebrei relaţionale

Proprietatile sunt explicate detaliat, fiind insotite de exemple, in documentul
Curs_7_Proprietati_Operatori_Exemple

Proprietăţile operatorilor algebrei relaţionale

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:

$\Pi_{A1,\ldots,Am}(\Pi_{B1,\ldots,Bn}(R)) = \Pi_{A1,\ldots,Am}(R)$,

unde $\{A_1, A_2,\ldots,A_m\} \subseteq \{B_1, B_2,\ldots,B_n\}$.

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.

Proprietăţile operatorilor algebrei relaţionale

Proprietatea 5. Comutarea selecţiei cu proiecţia:

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

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

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

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

Proprietăţile operatorilor algebrei relaţionale

Proprietatea 6. Comutarea selecţiei cu produsul cartezian:

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

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

Dacă condiţia este de forma $cond1 \wedge cond2$ şi dacă $cond1$ implică numai atribute din $R1$, iar $cond2$ implică numai atribute din $R2$, atunci

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

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

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

Proprietăţile operatorilor algebrei relaţionale

Proprietatea 7. Comutarea selecţiei cu reuniunea:

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

Proprietatea 8. Comutarea selecţiei cu diferenţa:

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

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:

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

Proprietatea 10. Comutarea proiecţiei cu reuniunea:

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

Proprietăţile operatorilor algebrei relaţionale

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.

Proprietăţile operatorilor algebrei relaţionale

Proprietatea 12. Compunerea selecţiei cu operaţia *join*:

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

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

Reguli de optimizare frecvent folosite

Regula de optimizare 1:

Selecţiile se execută cât mai devreme posibil. Motivaţia acestei reguli este că selecţiile reduc substanţial dimensiunea relaţiilor. Regula de transformare 4 poate fi folosită pentru a separa două sau mai multe selecţii în selecţii individuale care pot fi distribuite *join*-ului sau produsului cartezian folosind comutarea selecţiei cu *join*-ul.

Reguli de optimizare frecvent folosite

Regula de optimizare 2:

Produsele carteziene se înlocuiesc cu *join*-uri, ori de câte ori este posibil. Un produs cartezian între două relaţii este de obicei mult mai scump (ca şi cost) decât un *join* între cele două relaţii, deoarece primul generează concatenarea tuplurilor în mod exhaustiv şi poate genera un rezultat foarte mare. Această transformare se poate realiza folosind legătura dintre produs cartezian, *join* şi selecţie.

Reguli de optimizare frecvent folosite

Regula de optimizare 3:

Dacă sunt mai multe *join*-uri atunci cel care se execută primul este cel mai restrictiv

- Un *join* este mai restrictiv decât altul dacă produce o relaţie mai mică;
- Se poate determina care *join* este mai restrictiv pe baza selectivităţii sau cu ajutorul informaţiilor statistice;
- Algebric, acest lucru se poate realiza folosind regula de transformare 2;

Reguli de optimizare frecvent folosite

Regula de optimizare 4:

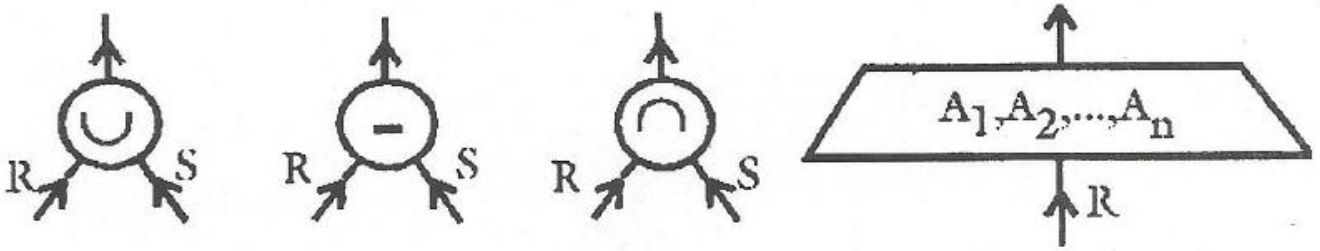
Proiecţiile se execută la început pentru a îndepărta atributele nefolositoare.

- Dacă un atribut al unei relaţii nu este folosit în operaţiile ulterioare atunci trebuie îndepărtat. În felul acesta se va folosi o relaţie mai mică în operaţiile ulterioare;
- Aceasta se poate realiza folosind comutarea proiecţiei cu *join*-ul;

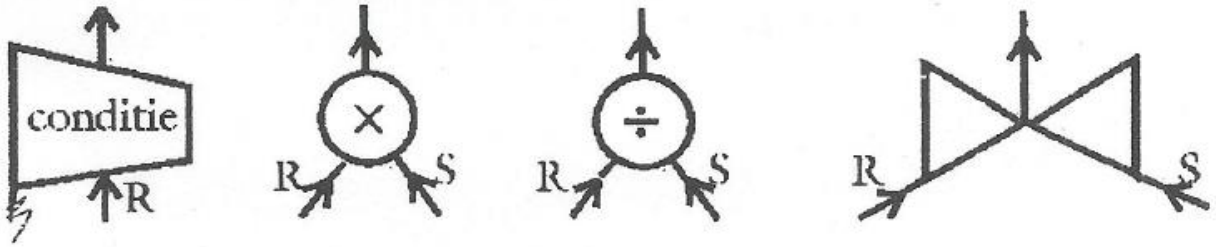
Arbori algebrici. Simbolurile operatorilor algebrei relationale.

- In continuare vom studia simbolurile operatorilor algebrei relationale;
- Cu ajutorul acestora, fiecare operator al algebrei relationale se poate reprezenta grafic;
- Reprezentarea grafica o sa formeze in final arborele algebric asociat expresiei algebrice

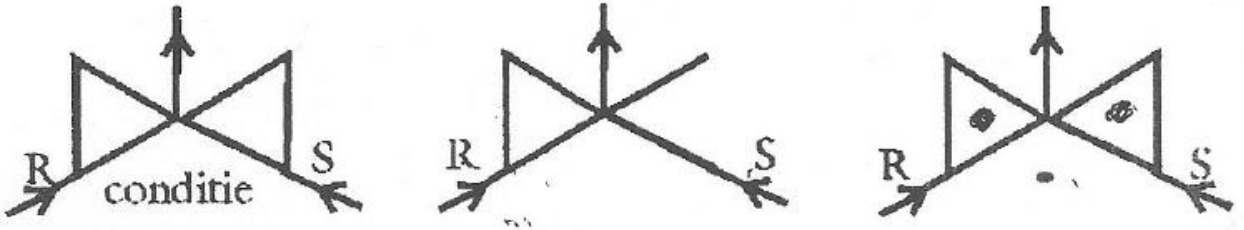
UNION DIFFERENCE INTERSECT PROJECT



SELECT PRODUCT DIVISION NATURAL JOIN

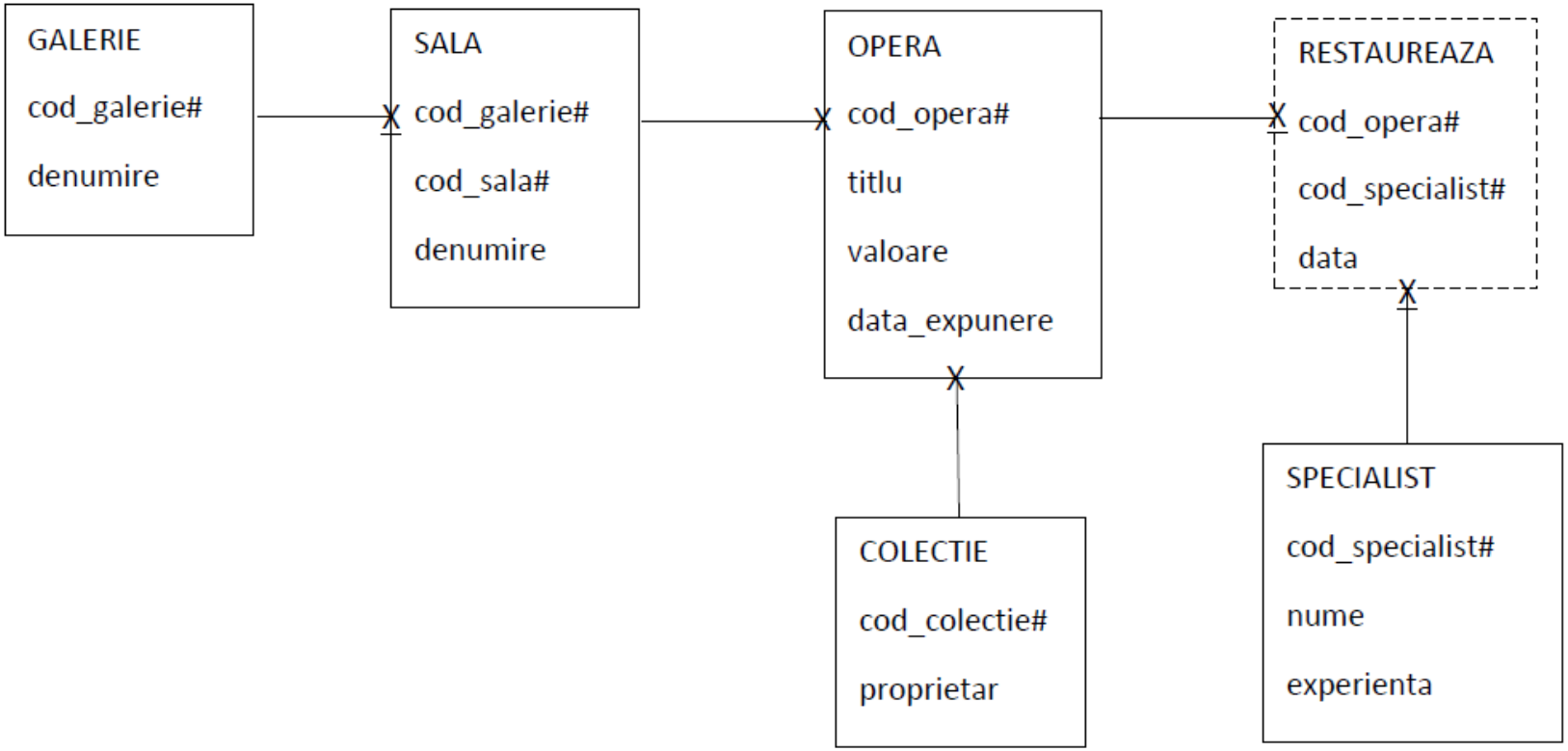


θ -JOIN SEMI-JOIN OUTER-JOIN



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.



Exemplu

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 upper(o.cod_galerie) = 'G1'
      AND initcap(c.proprietar) = 'King'
      AND r.data > to_date('05/06/2000', 'dd/mm/yyyy');
```

Exemplu

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)

Exemplu

Arbore algebric:

- Arborele algebric se afla pe slide-ul urmator, dar si in documentul numit: **Arbore_Algebric_Curs7**

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)