

Curs 9 : Algebra relationala

1. Definitii, clasificari
2. Operatorii algebrei relationale
3. Proprietatile operatorilor algebrei relationale
4. Optimizarea interogarilor

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Algebra relationala vs. calculul relational

2 moduri de exprimare a operațiilor specifice BD în modelul relational:

| <i>Algebra</i> | <i>Logic</i> |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| noile relații se obțin aplicând operatori specializați asupra uneia sau mai multor relații din cadrul bazei relaționale; | noile relații se obțin cu ajutorul unor formule logice pe care trebuie să le satisfacă tuplurile rezultatului; |
| algebra relațională (AR) este un limbaj formal procedural. | calculul relațional (CR) este un limbaj formal neprocedural, care utilizează predicate |



Observatie

J.D.Ullman: **AR** \equiv **CR**

i.e.: orice relație care poate fi definită în **AR** poate fi definită și în **CR** și reciproc;

1. Definiții, clasificări

2. Operatorii **AR**

3. Proprietatile operatorilor **AR**

4. Optimizarea interogărilor

Observatii

În abordarea algebrică:

- o relație = o mulțime de tupluri
- o BD = o mulțime de relații pe care sunt definiți operatorii algebrici;

Baza teoretică fundamentală pentru limbajele de interogare relaționale

≡ operatorii introduși de E.F. Codd pentru prelucrarea relațiilor

AR: proprietatea de închidere relațională:

i.e.: pot fi scrise expresii relaționale imbricate

(expresii relaționale în care operanzii sunt reprezentați tot prin expresii relaționale, de o complexitate arbitrară);

Operatorii **AR** permit exprimarea unor cereri nerecursive

în cazul unei cereri recursive: este necesar un operator special:

închiderea tranzitivă a unei relații.

1. Definiții, clasificări

2. Operatorii AR

3. Proprietatile operatorilor AR

4. Optimizarea interogărilor



Clasificari ale operatorilor relationali:

| <i>Prima clasificare</i> | <i>A doua clasificare</i> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> • mulțime <ul style="list-style-type: none"> √ intersecție, √ reuniune, √ diferență, √ produs cartezian • relaționali <ul style="list-style-type: none"> √ proiecție, √ selecție, √ diviziune, √ compunere; | <ul style="list-style-type: none"> • operatori ireductibili (de baza): <ul style="list-style-type: none"> √ selecție, √ proiecție, √ produs cartezian, √ reuniune √ diferență • operatori derivati: <ul style="list-style-type: none"> √ intersecție, √ diviziune √ compunere. |

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Funcțiile realizate de operatorii relationali:

1. **SELECT** (selecție) – extrage tupluri ce satisfac o condiție specificată;
2. **PROJECT** (proiecție) – extrage attributele specificate;
3. **DIFFERENCE** (diferență) – extrage tupluri care apar într-o relație, dar nu apar în cealaltă;
4. **PRODUCT** (produs cartezian) – generează toate perechile posibile de tupluri, primul element al perechii fiind luat din prima relație, iar cel de-al doilea element din cealaltă relație;
5. **UNION** (reuniune) – reunește două relații;
6. **INTERSECT** (intersecție) – extrage tupluri care apar în ambele relații;
7. **DIVISION** (diviziune) – extrage valorile atributelor dintr-o relație, care apar în toate valorile atributelor din cealaltă relație;

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Funcțiile realizate de operatorii relationali (cont.):

8. **JOIN** (compunere) – extrage tupluri din mai multe relații corelate:
9. **NATURAL JOIN** (compunere naturală) – combină tupluri din două relații, cu condiția ca attributele comune să aibă valori identice;
10. **SEMI-JOIN** (semi-compunere) – selectează tupluri doar dintr-o relație care vor fi corelate cu tuplurile celeilalte relații;
11. **Θ -JOIN** (Θ -compunere) – combină tupluri din două relații, cu condiția ca valorile atributelor specificate să satisfacă o anumită condiție;
12. **OUTER JOIN** (compunere externă) – combină tupluri din două relații, astfel încât condițiile de corelare să fie satisfăcute. Tuplurile din orice relație care nu satisfac aceste condiții sunt completate cu *null*.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor



Observatii

1. UNION, INTERSECT, DIFFERENCE:
 - se aplica numai la relații având aceeași aritate,
 - ordinea (nu numele) atributelor este aceeași;
2. Scopul fundamental al **AR**: scrierea expresiilor relaționale;
Aplicații posibile ale expresiilor relaționale:
 - definirea unui domeniu pentru interogare sau actualizare,
 - definirea constrângerilor de integritate și de securitate,
 - definirea datelor care vor fi incluse într-o vizualizare,
 - definirea datelor care vor reprezenta domeniul de valabilitate al unei operații de control al concurenței etc.

Curs 9 : Algebra relationala

1. Definitii, clasificari
2. **Operatorii algebrei relationale**
3. Proprietatile operatorilor algebrei relationale
4. Optimizarea interogarilor

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

1. Operatorul *PROJECT*



Proiecția =

= o operație unară care elimină anumite attribute ale unei relații R , producând o submulțime „pe verticală” a acesteia



Observatie

Suprimarea unor attribute poate avea ca efect apariția unor tupluri duplicate, care trebuie eliminate



Notatii

$\Pi_{A_1, \dots, A_m}(R),$

$PROJECT(R, A_1, \dots, A_m),$

$R[A_1, \dots, A_m],$

unde A_1, A_2, \dots, A_m sunt parametrii proiecției relativ la relația R
 $\equiv A_1, A_2, \dots, A_m$ sunt attributele din R care nu au fost eliminate prin proiecție (care apar în relația-rezultat R').

1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogărilor
-

1. Operatorul *PROJECT* (cont.)

 Exemplu:

Să se obțină numele și prenumele salariaților din FMI

Proiecție în algebra relațională:

Rezultat = PROJECT (PERSONAL_FMI, nume,
prenume)

Proiecție fără dubluri în SQL:

```
SELECT DISTINCT nume, prenume  
FROM          personal_fmi;
```

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

2. Operatorul *SELECT*



Selectia =

= o operație unară care elimină anumite tupluri ale unei relații R , producând o submulțime „pe orizontala” a acesteia



Observatii

Relatia R' rezultata din relatia R se obține prin extragerea tuplurilor din R care satisfac o condiție specificată

Condiția este o formulă logică ce poate cuprinde nume de attribute, constante, operatori logici, operatori aritmetici de comparare;



Notatii

SELECT (R , condiție),

R [condiție],

RESTRICT (R , condiție)

$\sigma_{\text{condiție}}(R)$.

1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogariilor
-

2. Operatorul **SELECT** (cont.)

 Exemplu:

Să se obțină toate informațiile despre cursurile optionale din FMI

Selectie în algebra relațională:

Rezultat = SELECT (CURS, tip='optional')

Selectie în SQL:

SELECT *

FROM curs

WHERE tip='optional';

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

3. Operatorul *UNION*



Reuniunea =

= fie S și T două relații de aceeași aritate; reuniunea lor, R , este tot o relație, care constă din mulțimea tuplurilor aparținând fie lui S , fie lui T , fie ambelor relații



Observatii

- ✓ Reuniunea este o operație binară comutativă;
- ✓ Operatorul de reuniune permite:
 - obținerea tuplurilor distincte a două relații
 - adăugarea de noi tupluri într-o relație;



Notatii

$UNION(S, T)$

$OR(S, T)$

$APPEND(S, T)$

$S \cup T$.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

3. Operatorul **UNION** (cont.)

 Exemplu:

Să se obțină lista completa a numelor si prenumelor cadrelor didactice si studentilor din FMI

Reuniune în algebra relațională:

$S = \text{PROJECT}(\text{CADRU_DIDACTIC}, \text{nume}, \text{prenume})$

$T = \text{PROJECT}(\text{STUDENT}, \text{nume}, \text{prenume})$

$\text{Rezultat} = \text{UNION}(S, T)$

Reuniune în SQL:

SELECT nume, prenume

FROM cadru_didactic

UNION

SELECT nume, prenume

FROM student;

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

4. Operatorul *DIFFERENCE*



Diferenta =

= fie S și T două relații de aceeași aritate; diferența lor, R , este tot o relație, care constă din mulțimea tuplurilor care aparțin lui S , dar nu aparțin lui T



Observatii

- ✓ Diferența este o operație binară NEcomutativă;
- ✓ Operatorul de diferență permite:
 - obținerea tuplurilor ce apar numai într-o relație
 - ștergerea tuplurilor dintr-o relație;



Notatii

DIFFERENCE (S, T)

MINUS (S, T)

REMOVE (S, T)

$S - T$.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

4. Operatorul *DIFFERENCE* (cont.)

⌚ Exemplu:

Să se obțină numele studentilor care nu se regăsesc printre numele angajaților din FMI

Diferența în algebra relațională:

$S = \text{PROJECT}(\text{STUDENT}, \text{nume})$

$T = \text{PROJECT}(\text{PERSONAL_FMI}, \text{nume})$

$\text{Rezultat} = \text{DIFFERENCE}(S, T)$

Diferența în SQL:

SELECT nume

FROM student

MINUS

SELECT nume

FROM personal_fmi;

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

5. Operatorul *INTERSECT*



Intersectia =

= fie S si T doua relatii de aceeaasi aritate; intersectia lor, R , este tot o relatie, care consta din multimea tuplurilor care aparțin atat lui S cat si lui T



Observatii

- ✓ Intersectia este o operatie binara comutativa;
- ✓ Operatorul de intersectie permite:
 - obținerea tuplurilor ce apar simultan in doua relatii
- ✓ Operatorul *INTERSECT* este un operator derivat:

$$S \cap T = S - (S - T)$$

$$S \cap T = T - (T - S).$$



Notatii

INTERSECT (S, T)

AND (S, T)

$S \cap T$.

1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogărilor
-

5. Operatorul **INTERSECT** (cont.)

⌚ Exemplu:

Să se obțină numele studentilor care coincid cu numele angajaților din FMI

Intersecția în algebra relațională:

$S = \text{PROJECT}(\text{STUDENT}, \text{nume})$

$T = \text{PROJECT}(\text{PERSONAL_FMI}, \text{nume})$

$\text{Rezultat} = \text{INTERSECT}(S, T)$

Intersecția în SQL:

```
SELECT nume
```

```
FROM student
```

```
INTERSECT
```

```
SELECT nume
```

```
FROM personal_fmi;
```

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

5. Operatorul **INTERSECT** (cont.)

Observatie

Operatorii INTERSECT și DIFFERENCE pot fi simulați în SQL (în cadrul comenzii SELECT) cu ajutorul opțiunilor EXISTS, NOT EXISTS, IN, != ANY.

Exemplu: simularea intersectiei in SQL

Să se obțină numele studentilor care coincid cu numele angajaților din FMI

```
SELECT  nume
FROM    student s
WHERE EXISTS
        (SELECT  nume
         FROM    personal_fmi p
         WHERE   s.nume = p.nume);
```

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

6. Operatorul *PRODUCT*



Produsul cartezian =

= fie S și T două relații de aritate m , respectiv n ; produsul cartezian al lui S cu T este tot o relație, fie ea R , care constă din mulțimea tuplurilor de aritate $m+n$ cu proprietatea că, în fiecare tuplu, primele m componente reprezintă un tuplu din S iar celelalte n componente reprezintă un tuplu din T



Observatii

- ✓ Produsul cartezian este o operație binară NEcomutativă;
- ✓ Este posibil ca cele două relații să aibă atribute cu același nume
=> pentru a menține unicitatea denumirilor atributelor din cadrul unei relații, denumirile acestor atribute se prefixează cu denumirea relației.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

6. Operatorul **PRODUCT** (cont.)

Notatii

PRODUCT(S, T)

TIMES (S, T)

,

$S \times T$

Exemplu:

Să se obțină lista tuturor posibilitatilor de alocare de cursuri
cadrelor didactice din FMI

Produs cartezian în algebra relațională:

Rezultat = PRODUCT (CADRU_DIDACTIC, CURS)

Produs cartezian în SQL:

SELECT *

FROM cadru_didactic, curs;

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

7. Operatorul *DIVISION*



Diviziunea =

= fie doua multimi de atribut:

$A = \{A_1, A_2, \dots, A_n\}$ și

$B = \{B_1, B_2, \dots, B_m\}$ si

doua relatii $S(A,B)$, de aritate $n+m$ si $T(B)$ de aritate m
(i.e.:multimea atributelor relatiei T este o submultime a
multimii atributelor relatiei S);

=> rezultatul aplicarii operatorului de diviziune asupra relatiilor
 S si T este o relatie R de aritate n cu proprietatea ca:

multimea atributelor sale coincide cu multimea de atribut A
(i.e.: consta din acele attribute care apartin relatiei S si nu
apartin relatiei T),

multimea tuplurilor sale rezulta prin selectarea tuplurilor din
relatia S astfel: fie $r[A] \in R \Rightarrow \exists t[B] \in T$ a.i $s[A,B] \in S$.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

7. Operatorul ***DIVISION*** (cont.)

$$R = S \div T = \Pi_A \sigma_{S.B=T.B}(S)$$

adica: intai se selecteaza un tuplu din S doar daca valorile atributelor sale B coincid cu valorile atributelor unui tuplu din T
 apoi, pe un astfel de tuplu, se aplica o proiectie pt a retine doar valorile atributelor din A;
 acestea formeaza un tuplu din R;

⌚ Exemplu:

| S | | | | | T | |
|---|---|----|---|---|---|---|
| A | | | B | | B | |
| 1 | 2 | 12 | m | s | m | a |
| 3 | 4 | 34 | f | e | y | e |
| 5 | 6 | 56 | a | t | a | t |

| R | | |
|---|---|----|
| A | | |
| 5 | 6 | 56 |

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

7. Operatorul *DIVISION* (cont.)



Notatii

DIVIDE (*S*, *T*)

DIVISION (*S*, *T*)

$S \div T$



Division = operator derivat:

$R = S \div T = \Pi_A \sigma_{S.B=T.B}(S)$ sau:

$S \div T = S_1 - S_2.$

unde: $S_1 = \Pi_{1,2,\dots,n}(S)$, $S_2 = \Pi_{1,2,\dots,n}((S_1 \times T) - S)$



Division = exprimare in SQL:

necesita utilizarea \forall (care nu exista in SQL!)

dar \forall poate fi simulat cu ajutorul \exists stiind ca:

$\forall x P(x) \equiv \neg \exists x \neg P(x)$

=> operatorul *DIVISION* poate fi exprimat în SQL prin succesiunea a doi operatori *NOT EXISTS*.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

7. Operatorul *DIVISION* (cont.)

⌚ Exemplu:

Să se obțină codurile studentilor care urmeaza cel puțin un curs facultativ

Diviziune în algebra relațională:

$S = \text{PROJECT (URMEAZA, codS, codC)}$

$T = \text{PROJECT (SELECT (CURS, tip='facultativ'), codC)}$

Rezultat = DIVISION (S, T) .

Diviziune în SQL:

```
SELECT  UNIQUE codS
FROM    urmeaza x
WHERE NOT EXISTS
  (SELECT *
   FROM  curs c
   WHERE curs.tip = 'facultativ'
   AND NOT EXISTS
     (SELECT *
      FROM  urmeaza b
      WHERE c.codC = b.codC
      AND   b.codS = x.codS));
```

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

7. Operatorul *DIVISION* (cont.)



Simularea diviziunii cu ajutorul funcției *COUNT*:

Aceeasi cerinta: să se obțină codurile studentilor care urmeaza cel puțin un curs facultativ

```
SELECT codS
FROM urmeaza
WHERE codC IN
      (SELECT codC
       FROM curs
       WHERE tip = 'facultativ')
GROUP BY codS
HAVING COUNT (codC) =
      (SELECT COUNT (*)
       FROM curs
       WHERE tip = 'facultativ');
```

- | | |
|---------------------------|-----------------------------------------|
| 1. Definiții, clasificări | 3. Proprietatile operatorilor AR |
| 2. Operatorii AR | 4. Optimizarea interogărilor |

Operatorul *JOIN*

= operator de **compunere** care permite regăsirea informației din mai multe relații corelate



Compunerea =

= operație binară asupra a 2 relații, S , T , care are ca rezultat o nouă relație, R , în care fiecare tuplu este o combinație a unui tuplu din S cu un tuplu din T



Observatii

1. Condiția necesară aplicării operatorului *JOIN*:
tuplurile care se combină să fie similare
2. Operatorul combina alți 3 operatori:
 - produsul cartezian,
 - selecția,
 - proiecția;
 În general:
 - se construiește un produs cartezian,
 - se elimină tupluri prin selecție,
 - se elimină attribute prin proiecție;
3. Există mai multe variante ale operatorului *JOIN*:
 - *NATURAL JOIN*,
 - *Θ -JOIN*
 - *SEMI-JOIN*,
 - *OUTER JOIN (LEFT, RIGHT, FULL)*.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul *NATURAL JOIN*

 Operatorul de compunere naturala =

= combină tupluri din două relații S și T , cu condiția ca attributele comune să aibă valori identice

 Notatii

,

$JOIN(S, T)$

$R \diamond S$

 Algoritmul

1. se calculează produsul cartezian $S \times T$,
2. pentru fiecare atribut comun A care definește o coloană în S și o coloană în T :

se selectează din $S \times T$ tuplurile ale căror valori coincid în coloanele $S.A$ și $T.A$ (atributul $S.A$ reprezintă numele coloanei din $S \times T$ corespunzătoare coloanei A din S),

3. pentru fiecare astfel de atribut A se proiectează coloana $T.A$, iar coloana $S.A$ se va numi A ;

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul **NATURAL JOIN** (cont.)

 NATURAL JOIN = operator derivat:

$$JOIN(S, T) = \Pi_{i_1, \dots, i_m} \sigma_{(S.A_1 = T.A_1) \wedge \dots \wedge (S.A_k = T.A_k)}(S \times T),$$

unde A_1, \dots, A_k = attributele comune lui S și T ,

i_1, \dots, i_m = lista componentelor din $S \times T$
(păstrând ordinea inițială) din care au fost eliminate
componentele $S.A_1, \dots, S.A_k$

 Exemplu:

Să se obțină informații complete despre studenții FMI și liceele pe care le-au absolvit, respectiv.

NATURAL JOIN în algebra relațională:

Rezultat = JOIN (STUDENT, LICEU)

NATURAL JOIN în SQL:

```
SELECT  *
FROM    student s, liceu l
WHERE   s.codL = l.codL;
```

1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogărilor
-


Operatorul θ -JOIN

 Operatorul de θ -compunere =

= combină tupluri din două relații S și T , cu condiția ca attributele menționate să îndeplinească o anumită condiție specificată explicit în cadrul operației

 Notatii

$JOIN(R, S, \text{conditie})$

 θ -JOIN = operator derivat: produs cartezian și selecție

$$JOIN(S, T, \text{conditie}) = \sigma_{\text{conditie}}(S \times T)$$

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul **θ -JOIN** (cont.)

 Exemplu:

Să se obțină informații despre studenții FMI (cod, nume, prenume, data și locul nașterii) și despre cadrele didactice (cod, nume, prenume, grad didactic, doctorat) cu condiția ca numele de familie ale cadrelor didactice și studenților să nu coincidă

Operatorul **θ -JOIN** în algebra relațională:

$S = \text{PROJECT} (\text{STUDENT}, \text{codS}, \text{nume}, \text{prenume}, \text{data_nastere}, \text{loc_nastere})$

$T = \text{PROJECT} (\text{CADRU_DIDACTIC}, \text{codCD}, \text{nume}, \text{prenume}, \text{gradD}, \text{doctorat})$

$\text{Rezultat} = \text{JOIN} (S; T, \text{STUDENT.nume} \neq \text{CADRU_DIDACTIC.nume})$

Operatorul **θ -JOIN** în SQL:


```
SELECT  codS, nume, prenume, data_nastere,
        loc_nastere, codCD, nume, prenume, gradD, doctorat
FROM    student s, cadru_didactic c
WHERE   s.nume <> c.nume;
```

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul **SEMI-JOIN**

 Operatorul de semi-compunere =

= generează o relație care conține toate tuplurile din S ce sunt corelate cu cel puțin unul dintre tuplurile din T


 Observatii

1. Operatorul este utilizat când nu sunt necesare toate atributele compunerii (sunt conservate atributele unei singure relații participante la compunere),
2. Operatorul este asimetric;

 Notatii

$SEMIJOIN(S, T)$

$SEMIJOIN(S, T, condiție)$.

 $SEMI-JOIN$ = operator derivat:


$SEMIJOIN(S, T) = \Pi_M(JOIN(S, T))$

$SEMIJOIN(S, T, condiție) = \Pi_M(JOIN(S, T, condiție))$,

unde M = mulțimea atributelor relației S .

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul *SEMI-JOIN* (cont.)

 Exemplu:

Să se obțină informații (nume, localitate) despre liceele ai caror absolvenți de altă naționalitate decât cea română au devenit studenți ai FMI

Operatorul *SEMI-JOIN* în algebra relațională:

$S = \text{SELECT (STUDENT, naționalitate } \neq \text{ 'română'})}$

$T = \text{JOIN (S, LICEU)}$

$\text{Rezultat} = \text{PROJECT (T, denumire, oras)}$

Operatorul *SEMI-JOIN* în SQL:

```
SELECT denumire, oras
FROM student s, liceu l
WHERE s.codL = l.codL
AND naționalitate <> 'română';
```

- | | |
|---------------------------|-----------------------------------------|
| 1. Definiții, clasificări | 3. Proprietatile operatorilor AR |
| 2. Operatorii AR | 4. Optimizarea interogărilor |

Operatorul **OUTER-JOIN**



Operatorul de compunere externa =

= combină tuplurile din două relații S și T pentru care sunt satisfăcute condițiile de corelare fără a pierde, însă, celelalte tupluri



Observatii

1. În cazul aplicării operatorului *JOIN* se pot pierde tupluri (există un tuplu în una din relații pentru care nu există niciun tuplu în cealaltă relație, astfel încât să fie satisfăcută relația de corelare)
 2. Operatorul *OUTER JOIN* elimină acest inconvenient astfel:
 - practic, se realizează compunerea naturala a două relații S și T
 - apoi se adaugă tuplurile din S și T , care nu sunt conținute în compunere, completate cu *null* acolo unde valorile atributelor există într-un tuplu din S (respectiv T) dar nu există și în T (respectiv S);
- ⇒ avantajul acestui operator:
se păstrează informațiile (i.e.: se păstrează tuplurile care ar fi fost pierdute în alte tipuri de *join*).

1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogărilor
-

Operatorul **OUTER-JOIN** (cont.)



Notatii

OUTERJOIN (S, T)

OUTERJOIN (S, T, condiție).




În practică apar trei tipuri de operatori OUTER JOIN:

- LEFT OUTER JOIN păstrează în rezultat fiecare tuplu al relației din stânga (aici: S)
- RIGHT OUTER JOIN păstrează în rezultat fiecare tuplu al relației din dreapta (aici: T)
- FULL OUTER JOIN păstrează tuplurile din ambele relații, completate cu *null* atunci când nu există tupluri corelate.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Operatorul **OUTER-JOIN** (cont.)

 Exemplu:

Să se obțină informații complete referitoare la studenții FMI și la cursurile optionale urmate de aceștia, sesizând cazurile în care există cursuri optionale la care nu s-a înscris niciun student, respectiv studenți care nu s-au înscris la niciun curs optional

Operatorul *OUTER JOIN* în algebra relațională:

Rezultat = OUTERJOIN (STUDENT, CURS)

Operatorul *OUTER JOIN* în SQL:

```
SELECT *
FROM student s FULL OUTER JOIN
  (SELECT *
   FROM curs c
   WHERE c.tip = 'optional')
ON (s.codS = c.codC);
```


1. Definiții, clasificări
 2. Operatorii **AR**
 3. Proprietatile operatorilor **AR**
 4. Optimizarea interogărilor
-

Concluzii

- ✓ **AR** este o colecție de operații asupra relațiilor.
- ✓ Cinci dintre cele 8 operații propuse de E.F.Codd:
 - proiecția,
 - selecția,
 - reuniunea,
 - diferența,
 - produsul carteziansunt operații primitive și constituie mulțimea minimă de operații ale **AR**.
- ✓ Pe baza celor 5 operații primitive:
 - se poate propune o definiție alternativă a celor trei operatori (numiți derivați):
 - ☐ intersecția = o diferență recursivă de relații
 - ☐ diviziunea = o proiecție a unei selecții asupra relației de împărțit.
 - ☐ joncțiunea = o proiecție a unei selecții a produsului cartezian al celor două relații,
 - se poate construi orice expresie a algebrei relationale.

1. Definiții, clasificări
2. Operatorii **AR**
3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Concluzii (cont.)

Pentru prelucrarea optimă a relațiilor au fost introduse operații adiționale:



complement =

permite determinarea complementului unei relații, permițând găsirea tuplurilor care nu aparțin unei relații;



despicare =

permite ruperea, în funcție de o condiție, a unei relații în două, dintre care doar una satisface condiția;



închidere tranzitivă =

permite adăugarea de noi tupluri unei relații, tupluri deduse prin tranzitivitate;



funcții asociate =

permit calcularea maximului (MAX), minimului (MIN), mediei aritmetice (AVG), dispersiei (VAR) etc..

Curs 9 : Algebra relationala

1. Definitii, clasificari
2. Operatorii algebrei relationale
3. **Proprietatile operatorilor algebrei relationale**
4. Optimizarea interogarilor

1. Definiții, clasificări
2. Operatorii **AR**
3. **Proprietatile operatorilor AR**
4. Optimizarea interogărilor

Proprietatile operatorilor AR

 **Expresie a algebrei relaționale =**

o expresie în care:

- operanzii = relații (în sensul lui E.F. Codd),
- operatorii =
 - cei 8 operatori (primitivi sau derivați) ai **AR** (proiecție, selecție, etc.), plus, eventual,
 - operatorii suplimentari (complement, despicare, închidere tranzitivă) și
 - funcțiile asociate (MIN, MAX, AVG, VAR);

 Două expresii sunt **echivalente** \Leftrightarrow

în urma evaluării lor, se obține ca rezultat aceeași relație.

1. Definiții, clasificări
2. Operatorii **AR**

3. **Proprietatile operatorilor AR**
4. Optimizarea interogărilor

Proprietatile operatorilor AR



Observatii

- O expresie **AR** \cong un plan de executie a cererii;
- 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:
 - efectuarea prelucrărilor indicate de operatori
 - în ordinea aparițiilor acestora, sau
 - în ordinea fixată prin paranteze;
- Rezultatul evaluării unei expresii:
 - o relație derivată din relațiile menționate ca operanzi în cadrul expresiei;
- Ordinea în care se efectuează operațiile:
 - rolul cel mai important în evaluarea costului necesar realizării interogării.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

Proprietatile operatorilor **AR** (cont.)

- 🔔 2 metode de determinare a ordinii optime de execuție a operațiilor dintr-o expresie **AR**:
 1. algebric,
 2. prin estimarea costului:
- (1) Optimizarea cererilor bazată pe **AR** :
 - se exprimă cererile sub forma unor expresii algebrice relaționale,
 - se aplică transformări algebrice care conduc la expresii echivalente, dar care vor fi executate mai eficient;
aceste transformări au la baza o strategie de optimizare:
 - independentă de modul de memorare a datelor (strategie generală),
 - dependentă de modul de memorare (strategie specifică unui anumit SGBD);
- 🔔 Regulile de transformare a unui plan de execuție / expresie **AR**
≡ proprietăți ale operatorilor algebrici care permit ordonarea într-o altă formă, mai convenabilă, a operațiilor din interogare.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

Proprietatile operatorilor AR (cont.)

Proprietatea 1. Operațiile join și produs cartezian: comutative

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

$$R1 \times R2 = R2 \times R1$$

Proprietatea 2. Operațiile join și produs cartezian: asociative

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

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

Proprietatea 3. Compunerea proiecțiilor:

$$\Pi_{A_1, \dots, A_m} (\Pi_{B_1, \dots, B_n} (R)) = \Pi_{A_1, \dots, A_m} (R)$$

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

Proprietatea 4. Compunerea selecțiilor:

$$\sigma_{cond1} (\sigma_{cond2} (R)) = \sigma_{cond1 \wedge cond2} (R) = \sigma_{cond2} (\sigma_{cond1} (R))$$

unde am notat prin *cond* condiția după care se face selecția.

1. Definiții, clasificări
2. Operatorii **AR**

3. **Proprietatile operatorilor AR**
4. Optimizarea interogurilor

Proprietatile operatorilor AR (cont.)

Proprietatea 5. Comutarea selecției cu proiecția:

dacă *cond* implică numai attributele A_1, \dots, A_m atunci:

$$\Pi_{A_1, \dots, A_m} (\sigma_{cond} (R)) = \sigma_{cond} (\Pi_{A_1, \dots, A_m} (R))$$

dacă *cond* implică și attributele B_1, \dots, B_n , care nu aparțin mulțimii $\{A_1, \dots, A_m\}$

$$\Pi_{A_1, \dots, A_m} (\sigma_{cond} (R)) = \Pi_{A_1, \dots, A_m} (\sigma_{cond} (\Pi_{A_1, \dots, A_m, B_1, \dots, B_n} (R)))$$

Proprietatea 6. Comutarea selecției cu produsul cartezian:

dacă *cond* implică numai attribute ale relației *R1* atunci:

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

dacă *cond* = *cond1* \wedge *cond2*

cond1 implică numai attribute din *R1*

cond2 implică numai attribute din *R2*, atunci:

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

dacă *cond1* implică numai attribute din *R1*

cond2 implică attribute atât din *R1* cât și din *R2*, atunci:

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

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

Proprietatile operatorilor AR (cont.)

Proprietatea 7. Comutarea selecției cu reuniunea:

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

Proprietatea 8. Comutarea selecției cu diferența:

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

Proprietatea 9. Comutarea proiecției cu reuniunea:

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

Proprietatea 10. Comutarea proiecției cu produsul cartezian:

dacă $\{A_1, \dots, A_m\} = \{B_1, \dots, B_n, C_1, \dots, C_k\}$

unde B_1, \dots, B_n sunt attribute ale relației $R1$

C_1, \dots, C_k sunt attribute ale relației $R2$ atunci:

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

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

Proprietatile operatorilor AR (cont.)

Proprietatea 11. Compunerea proiecției cu operația join:

dacă $\{A_1, \dots, A_m\} = \{B_1, \dots, B_n, C_1, \dots, C_k\}$

unde B_1, \dots, B_n sunt attribute ale relației $R1$

C_1, \dots, C_k sunt attribute ale relației $R2$ atunci:

$$\Pi_{A_1, \dots, A_m} (JOIN(R1, R2, D)) =$$

$$\Pi_{A_1, \dots, A_m} (JOIN(\Pi_{D, B_1, \dots, B_n}(R1), \Pi_{D, C_1, \dots, C_k}(R2), D),$$

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

Proprietatea 12. Compunerea selecției cu operația join:

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

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

unde A reprezintă mulțimea de attribute care apar în $cond$.

Curs 9 : Algebra relationala

1. Definitii, clasificari
2. Operatorii algebrei relationale
3. Proprietatile operatorilor algebrei relationale
4. **Optimizarea interogarilor**

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogarilor

Optimizarea interogarilor

Obiectivul optimizarii:

creșterea vitezei de access la informația din BD \equiv
 \equiv reducerea timpului de acces



Evident: interogările care necesita cel mai lung timp de prelucrare: interogările care contin operatorii:

- produs cartezian,
- compunere;


Cele mai utilizate metode :

1. reducerea timpului de răspuns (\equiv timpul total de execuție a interogării \equiv suma timpilor de execuție pentru fiecare dintre operațiile elementare care compun interogarea):
 - prin minimizarea timpului de executie al fiecărei operatii elementare care compune introgarea,
 - prin maximizarea numărului de operații paralele;
2. metoda euristica:
 - are in vedere ordinea de executie a operatiilor **AR**,
 - se bazeaza pe cele 12 proprietati enumerate mai sus,

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. **Optimizarea interogărilor**

Optimizarea interogărilor (cont.)

-  Ambele metode depind de informațiile statistice despre BD (ex.: în dicționarul datelor pot fi stocate informații statistice referitoare la:
- cardinalitatea relațiilor,
 - numărul de blocuri necesare pentru stocarea unei relații,
 - numărul de tupluri dintr-o relație care intră într-un singur bloc,
 - numărul de niveluri dintr-un index,
 - numărul de valori distincte pentru fiecare atribut etc.)
- => reactualizarea: cand sistemul este cu activitate redusă, și nu de fiecare dată când este inserat, șters sau reactualizat un tuplu.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Optimizarea interogărilor (cont.)

 Examinarea celor 12 proprietati:

⇒ conturarea strategiei de optimizare (de reordonare a operațiilor din cereri):

- efectuarea mai întâi a operațiilor unare (selectii înainte de proiectii) - care reduc dimensiunea relațiilor - și apoi a operațiilor binare (⇒ coborârea selecțiilor și proiecțiilor cât mai jos posibil în arborele algebric),
- regruparea compunerilor de selecții și proiecții într-o selecție urmată de o proiecție,
- regruparea selecțiilor și proiecțiilor “in cascadă”, prin unul dintre operatorii algebrici binari,
- combinarea proiecțiilor cu operațiuni binare adiacente,
- combinarea unora din selecții cu produse carteziane care eventual le preced, pentru a obține compuneri,
- înainte de compuneri, prelucrarea preliminară a fișierelor (relațiilor) prin operațiuni de sortare și de indexare,
- căutarea în expresiile mai complexe a sub-expresiilor care se repetă.


1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor


Optimizarea interogărilor (cont.)

Formal: 4 reguli euristice de optimizare:


Regula de optimizare 1. Selecțiile se execută cât mai devreme posibil pt ca reduc substanțial dimensiunea relațiilor;


 Proprietatea 4 poate fi folosită pentru a separa două sau mai multe selecții în selecții individuale :

$$\sigma_{cond1} (\sigma_{cond2} (R)) = \sigma_{cond1 \wedge cond2} (R) = \sigma_{cond2} (\sigma_{cond1} (R))$$

 aceste selectii individuale pot fi distribuite *join*-ului sau produsului cartezian folosind comutarea selecției cu *join*-ul

Regula de optimizare 2. Produsele carteziane se înlocuiesc cu *join*-uri, ori de câte ori este posibil

 un produs cartezian între două relații generează toate combinațiile de tupluri din cele 2 relații (i.e. un cardinal f mare)

 această transformare se poate realiza folosind legătura dintre produs cartezian, *join* și selecție.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogărilor

Optimizarea interogărilor (cont.)

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 relație mai mică

🔔 se poate determina care *join* este mai restrictiv:

- pe baza factorului de selectivitate
- cu ajutorul informațiilor statistice
- cu Proprietatea 2: asociativitatea operației de *join*:
$$JOIN(JOIN(R1, R2), R3) = JOIN(R1, JOIN(R2, R3))$$

Regula de optimizare 4. Proiecțiile se execută la început pentru a îndepărta attributele nefolositoare.

.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogarilor

Optimizarea interogarilor (cont.)

Algoritm pentru optimizarea expresiilor relationale

Intrare: o expresie relationala, reprezentata printr-un arbore sintactic

Ieșire: o expresie relationala optimizata, reprezentata tot printr-un arbore sintactic

Metoda:

Pas 1. Fiecare selectie $\sigma_{cond1 \wedge cond2 \wedge \dots \wedge condn} (R)$ este transformată în secvența $\sigma_{cond1} (\sigma_{cond2} (\dots (\sigma_{condn} (R))))$ (*Proprietatea 4*)

Pas 2. Fiecare selecție este deplasată cât mai jos posibil în arborele sintactic (*Proprietatile 4-7*)

Pas 3. Fiecare proiecție este deplasată cât mai jos posibil în arborele sintactic (*Proprietatile 3, 10, 9, 5*)

Pas 4. Secvențele de selecții și proiecții sunt combinate în:

- selecții unice,
- proiecții unice, sau
- selecții urmate de proiecții (regulile 3, 4, 5)



Atentie: operatiile Pasului 4, pot încălca principiul potrivit căruia proiecțiile sunt efectuate cât mai curând posibil

⇒ trebuie analizat castigul fiecărei variante: Pasul 4 sau Propr. 5!

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. **Optimizarea interogarilor**

Optimizarea interogarilor (cont.)



Algoritm pentru optimizarea expresiilor relationale (cont.)

Pas 5. Nodurile interne ale arborelui rezultate prin parcurgerea pașilor anteriori sunt grupate în „blocuri”

- Fiecare nod intern care corespunde unei operațiuni binare poate face parte din același bloc ca și predecesorii săi imediați cu care sunt asociate operațiuni unare
- Din bloc poate face parte și orice lanț de noduri succesoare asociate cu operațiuni unare și terminate cu o frunză



Ultima regulă nu se aplică atunci când operația binară este un produs cartezian neurmat de o selecție care sa se combine cu produsul menționat, astfel încât să formeze o θ -compunere!

Pas 6. Se evaluează fiecare bloc, în orice ordine, astfel încât niciunul din blocuri nu este evaluat înaintea grupurilor sale succesoare.

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogurilor

Optimizarea interogurilor (cont.)

⌘ Exemplu.

Fie o baza de date cu entitatile:

CIRCUIT (*Cnume*, *Fnume*, *Cod*),

FURNIZOR (*Fnume*, *Fadr*),

UTILIZATOR (*Unume*, *Uadr*, *Nrdoc*),

LIVRARI (*Nrdoc*, *Cod*, *Data*);

Pp că pentru a returna anumite informatii privind livrările de circuite este construită mai întâi o vizualizare care conține date referitoare la circuitele livrate

← se utilizeaza relațiile LIVRARI, UTILIZATOR, CIRCUIT;

Vizualizarea va fi definită prin expresia relațională:

$$\Pi_V(\sigma_U(\text{LIVRARI} \times \text{UTILIZATOR} \times \text{CIRCUIT})),$$

unde:

$$V = \{Cnume, Fnume, Cod, Unume, Uadr, Nrdoc, Data\}$$

$$U = \text{UTILIZATOR.Nrdoc} = \text{LIVRARI.Nrdoc} \wedge \text{CIRCUIT.Cod} = \text{LIVRARI.Cod.}$$

1. Definiții, clasificări
2. Operatorii **AR**

3. Proprietatile operatorilor **AR**
4. Optimizarea interogarilor

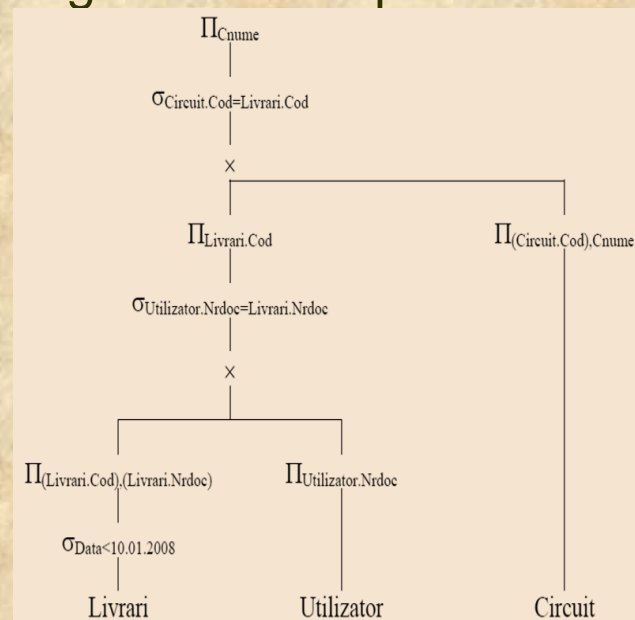
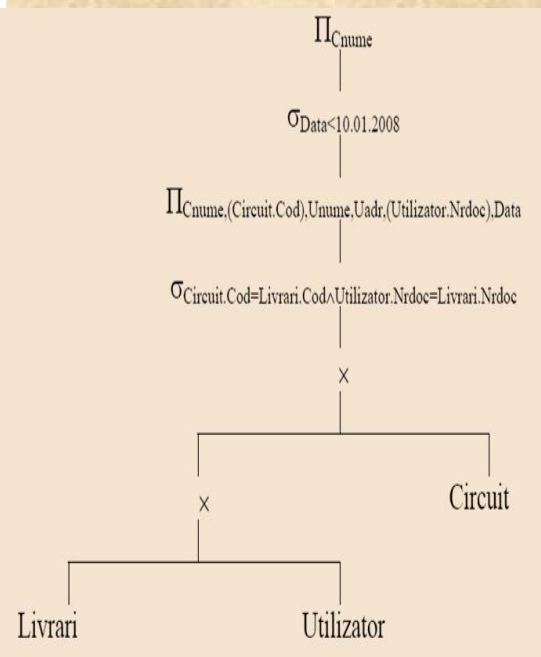
Optimizarea interogarilor (cont.)

Pp că interogarea solicită lista numelor circuitelor livrate înainte de 14 februarie 2014

⇒ În termenii algebrei relaționale:

$\Pi_{Cnume} (\sigma_{Data < 14.02.2014} (\Pi_U (\sigma_V (LIVRARI \times UTILIZATOR \times CIRCUIT))))$

Pentru evaluarea expresiei: se construiește arborele ei sintactic
se aplica algoritmul de optimizare:



Curs 9 : Algebra relationala

1. Definitii, clasificari
2. Operatorii algebrei relationale
3. Proprietatile operatorilor algebrei relationale
4. Optimizarea interogarilor