



BAZE DE DATE

CURS 6

ALGEBRA RELAȚIONALĂ

- **LDD** – precizează **entitățile**, relațiile dintre ele, **atributele**, structura atributelor, **cheile**, **constrângerile**, prin urmare definește **structura obiectelor bazei de date** (**schema** bazei).
- **LMD** – cuprinde aspecte referitoare la introducerea, eliminarea, modificarea și căutarea datelor.

ALGEBRA RELAȚIONALĂ

- Modelul relațional oferă **două mulțimi de operatori pe relații**:
 - **algebra relațională** (filtrele se obțin aplicând operatori specializați asupra uneia sau mai multor relații din cadrul bazei relaționale);
 - **calculul relațional** (filtrele se obțin cu ajutorul unor formule logice pe care tuplurile rezultatului trebuie să le satisfacă).
- Echivalența dintre algebra relațională și calculul relațional a fost demonstrată de J.D.Ullman. Această echivalență arată că orice relație posibil de definit în algebra relațională poate fi definită și în cadrul calcului relațional, și reciproc.

ALGEBRA RELAȚIONALĂ

- **Algebra relațională** a fost introdusă de E.F. Codd
 - mulțime de operații formale acționând asupra unor relații și având ca rezultat alte relații.
- Baza teoretică pentru limbajele de interogare relaționale o constituie operatorii introduși de Codd pentru prelucrarea relațiilor.
- **Operatorii sunt numai pentru citire** (nu actualizează operanți)!!!

ALGEBRA RELAȚIONALĂ

- Scopul fundamental al algebrei relaționale este de a permite scrierea **expresiilor relaționale**.
 - **reprezentare** de nivel superior, simbolică, a intențiilor utilizatorului și pot fi supuse unei diversități de **reguli de transformare** (**optimizare**).
- Relațiile sunt închise față de algebra relațională
 - operanzii și rezultatele sunt relații → ieșirea unei operații poate deveni intrare pentru alta → posibilitatea imbricării expresiilor în algebra relațională.

ALGEBRA RELAȚIONALĂ

- **Operatorii algebrei relaționale** sunt:
 - operatori tradiționali pe mulțimi (**UNION, INTERSECT, PRODUCT, DIFFERENCE**);
 - operatori relaționali speciali (**PROJECT, SELECT, JOIN, DIVISION**).

ALGEBRA RELAȚIONALĂ

Operatorul PROJECT

- Proiecția este o operație unară care elimină anumite attribute ale unei relații producând o submulțime „pe verticală” a acesteia.
 - Suprimarea unor attribute poate avea ca efect apariția unor tupluri duplicate, care trebuie eliminate.
- **Notatii:**
 - $\Pi_{A_1, \dots, A_m} (R)$
 - $\text{PROJECT} (R, A_1, \dots, A_m)$
 - $R[A_1, \dots, A_m]$

unde A_1, A_2, \dots, A_m sunt parametrii proiecției relativ la relația R .

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină o listă ce conține numele, prenumele și *job*-ul angajaților.

1. Proiecție în algebra relațională:

Rezultat = **PROJECT** (SALARIAT, nume, prenume, job)

2. Proiecție cu dubluri în SQL:

```
SELECT  nume, prenume, job
FROM    salariat;
```

3. Proiecție fără dubluri în SQL:

```
SELECT  DISTINCT nume, prenume, job
FROM    salariat;
```


ALGEBRA RELAȚIONALĂ

Operatorul SELECT

- Selecția (restricția) este o operație unară care produce o submulțime pe „orizontală” a unei relații R .
 - Această submulțime se obține prin extragerea tuplurilor din R care satisfac o condiție specificată.
- **Notatii:**
 - $\sigma_{\text{condiție}}(R)$
 - $R[\text{condiție}]$
 - **SELECT**(R , condiție)
 - **RESTRICT**(R , condiție).

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină informații complete despre programatori.

1. Selecție în algebra relațională:

Rezultat = **SELECT** (SALARIAT, job = 'programator')

2. Selecție în SQL:

```
SELECT  *  
FROM    salariat  
WHERE   job = 'programator';
```

ALGEBRA RELAȚIONALĂ

Operatorul UNION

- Reuniunea a două relații R și S este mulțimea tuplurilor aparținând fie lui R , fie lui S , fie ambelor relații.
- **Notății:**
 - $R \cup S$
 - $\text{UNION}(R, S)$
 - $\text{OR}(R, S)$
 - $\text{APPEND}(R, S)$.

Exemplu. Să se obțină lista cu numele persoanelor fizice și a subantreprenorilor.

```
SELECT  nume
FROM subantreprenor
UNION
SELECT  nume
FROM pers_fizica;
```

ALGEBRA RELAȚIONALĂ

Operatorul DIFFERENCE

- Diferența a două relații R și S este mulțimea tuplurilor care aparțin lui R , dar nu aparțin lui S .
- Diferența este o operație binară necomutativă care permite obținerea tuplurilor ce apar numai într-o relație.
- **Notatii:**
 - $R - S$
 - $\text{DIFFERENCE}(R, S)$
 - $\text{REMOVE}(R, S)$
 - $\text{MINUS}(R, S)$.

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină lista cu numărul contractului, tipul contractului, valoarea investiției și durata lucrării pentru contractele de subantrepriză pentru care valoarea investiției nu depășește 60000\$.

1. Diferență în algebra relațională:

$R = \text{PROJECT} (\text{SELECT} (\text{CONTRACT}, \text{tip_contract} = 'T'), \text{nr_contract}, \text{tip_contract}, \text{val_investitie}, \text{durata_lucrare});$

$S = \text{PROJECT} (\text{SELECT} (\text{CONTRACT}, \text{val_investitie} > 60000), \text{nr_contract}, \text{tip_contract}, \text{val_investitie}, \text{durata_lucrare});$

Rezultat = **DIFFERENCE** (R, S)

ALGEBRA RELAȚIONALĂ

2. Diferența în SQL:

```
SELECT  nr_contract,tip_contract,
        val_investitie,durata_lucrare
FROM    contract
WHERE   tip_contract = 'T'
MINUS
SELECT  nr_contract,tip_contract,
        val_investitie,durata_lucrare
FROM    contract
WHERE   val_investitie>60000;
```

Evident diferența se poate referi la tabele diferite! Implementați cererea prin care se listează toate orașele în care se află cel puțin o filială, dar nici o proprietate.

ALGEBRA RELAȚIONALĂ

Operatorul INTERSECT

- Intersecția a două relații R și S este mulțimea tuplurilor care aparțin atât lui R , cât și lui S . Operatorul INTERSECT este un operator binar, comutativ, derivat:
 - $R \cap S = R - (R - S)$
 - $R \cap S = S - (S - R)$.
- **Notății:**
 - **INTERSECT(R, S)**
 - $R \cap S$
 - **AND(R, S).**
- În anumite dialecte SQL există operator special (INTERSECT), care realizează această operație.
- Operatorii INTERSECT și DIFFERENCE pot fi simulați în SQL (în cadrul comenzii SELECT) cu ajutorul opțiunilor EXISTS, NOT EXISTS, IN, != ANY.

ALGEBRA RELAȚIONALĂ

Exemplu. Utilizând tabelele *angajati* și *departamente*, să se obțină lista codurilor salariaților care sunt directori de departament, dar și șefi direcți ai altor persoane.

1. Intersecție în algebra relațională:

$R = \text{PROJECT}(\text{ANGAJATI}, \text{cod_sef});$

$S = \text{PROJECT}(\text{DEPARTAMENTE}, \text{cod_director}),$

Rezultat = **INTERSECT** (R, S).

2. Intersecție în SQL:

```
SELECT cod_sef
FROM angajati
INTERSECT
SELECT cod_director
FROM departamente;
```

Simularea \cap cu op. in :

```
SELECT distinct cod_sef
FROM angajati
WHERE cod_sef IN (
  (SELECT cod_dir FROM dep.)
);
```

subacare
memorată
↓
nulează

ALGEBRA RELAȚIONALĂ

3. Simularea intersecției în SQL:

```
SELECT  cod_director
FROM    departamente d
WHERE   EXISTS
  (SELECT cod_sef
   FROM  angajati a
   WHERE d.cod_director=a.cod_sef);
```



← intersecție microinvenții

over $\{Buc, null\}$
 $\Leftrightarrow over \neq 'Buc' \wedge over \neq null$
 $T \wedge null = null$

- Cum simulăm intersecția utilizând operatorul IN?
- Cum simulăm diferența de la exercițiul de pe slide-ul 14?

NOT IN
 select over from filiale
 where over NOT IN (
 {Buc, Pitesti, Blaj}

select over
 from prop. in ch.
 {Buc, null}

NOT EXISTS → după a lab
 subscrieri
 → rezultat = null
 Sol 1: NVL(over, 'No null')
 Sol 2: where over IS NOT null;

ALGEBRA RELAȚIONALĂ

Operatorul PRODUCT

- Fie R și S relații de aritate m , respectiv n . Produsul cartezian al lui R cu S este mulțimea tuplurilor de aritate $m + n$ unde primele m componente formează un tuplu în R , iar ultimele n componente formează un tuplu în S .
- **Notatii:**
 - $R \times S$
 - $\text{PRODUCT}(R, S)$
 - $\text{TIMES}(R, S)$.

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină lista tuturor posibilităților de investiție în diverse obiective de către o firmă care este persoană juridică.

1. Produs cartezian în algebra relațională:

$R = \text{PROJECT}(\text{PERS_JURIDICA}, \text{nume}, \text{cod_contractant});$

$S = \text{PROJECT}(\text{OBIECTIV_INVESTITIE}, \text{denumire});$

Rezultat = $\text{PRODUCT}(R, S).$

2. Produs cartezian în SQL:

```
SELECT  cod_contractant, nume, denumire
FROM    obiectiv_investitie, pers_juridica;
```

ALGEBRA RELAȚIONALĂ

Operatorul DIVISION

- Diviziunea este o operație binară care definește o relație ce conține valorile atributelor dintr-o relație care apar **în toate** valorile atributelor din cealaltă relație.
- **Notatii:**
 - $DIVIDE(R, S)$
 - $DIVISION(R, S)$
 - $R \div S$.
- Diviziunea conține acele tupluri de dimensiune $n - m$ la care, adăugând orice tuplu din S , se obține un tuplu din R .
- Operatorul diviziune poate fi exprimat formal astfel:

$$R^{(n)} \div S^{(m)} = \{t^{(n-m)} \mid \forall s \in S, (t, s) \in R\}, \text{ unde } n > m \text{ și } S \neq \emptyset.$$

ALGEBRA RELAȚIONALĂ

- Operatorul DIVISION este legat de cuantificatorul universal (\forall) care nu există în SQL.
- Cuantificatorul universal poate fi însă simulat cu ajutorul cuantificatorului existențial (\exists) utilizând relația:

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

- Prin urmare, operatorul DIVISION poate fi exprimat în SQL prin succesiunea a doi operatori NOT EXISTS.

ALGEBRA RELAȚIONALĂ

Exemplu.

$R(A, B, C, D); S(C, D)$

A	B	C	D
a	b	x	y
a	b	z	t
a	c	x	y
c	a	x	t
c	d	x	y
c	d	z	t

C	D
x	y
z	t

DIVISION(R, S) = ?

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină codurile salariaților atașați **tuturor** proiectelor pentru care s-a alocat un buget egal cu 1000.

1. Diviziune în algebra relațională:

```
R = PROJECT(ATASAT_LA, cod_salariat, nr_proiect);
```

```
S = PROJECT(SELECT(PROIECT, buget = 1000), nr_proiect);
```

```
Rezultat = DIVISION(R, S).
```

2. Diviziune în SQL:

```
SELECT UNIQUE cod_salariat
FROM atasat_la aa
WHERE NOT EXISTS
  (SELECT *
   FROM proiect pp
   WHERE buget=1000
   AND NOT EXISTS
     (SELECT *
      FROM atasat_la bb
      WHERE pp.nr_proiect=bb.nr_proiect
      AND bb.cod_salariat=aa.cod_salariat));
```

ALGEBRA RELAȚIONALĂ

3. Simularea diviziunii cu ajutorul funcției COUNT:

```
SELECT  cod_salariat
FROM    atasat_la
WHERE   nr_proiect IN
        (SELECT nr_proiect
         FROM    proiect
         WHERE   buget=1000)
GROUP BY  cod_salariat
HAVING   COUNT(nr_proiect)=
        (SELECT COUNT(*)
         FROM    proiect
         WHERE   buget=1000);
```


ALGEBRA RELAȚIONALĂ

Operatorul JOIN

- Operatorul de compunere (uniune) permite regăsirea informației din mai multe relații corelate.
- Operatorul combină produsul cartezian, selecția și proiecția.
- 4 tipuri de join:
 - NATURAL JOIN
 - θ -JOIN
 - SEMI-JOIN
 - OUTER JOIN

ALGEBRA RELAȚIONALĂ

NATURAL JOIN

- Operatorul de compunere naturală (NATURAL JOIN) combină tupluri din două relații R și S , cu condiția ca atributele comune să aibă valori identice.
- Algoritmul care realizează compunerea naturală este următorul:
 1. se calculează produsul cartezian $R \times S$;
 2. pentru fiecare atribut comun A care definește o coloană în R și o coloană în S , se selectează din $R \times S$ tuplurile ale căror valori coincid în coloanele $R.A$ și $S.A$ (atributul $R.A$ reprezintă numele coloanei din $R \times S$ corespunzătoare coloanei A din R);
 3. pentru fiecare astfel de atribut A se elimină coloana $S.A$, iar coloana $R.A$ se va numi A .
- Operatorul NATURAL JOIN poate fi exprimat formal astfel:
$$\text{JOIN}(R, S) = \Pi_{i_1, \dots, i_m} \sigma_{(R.A_1 = S.A_1) \wedge \dots \wedge (R.A_k = S.A_k)}(R \times S),$$
unde A_1, \dots, A_k sunt atributele comune lui R și S , iar i_1, \dots, i_m reprezintă lista componentelor din $R \times S$ (păstrând ordinea inițială) din care au fost eliminate componentele $S.A_1, \dots, S.A_k$.

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină informații complete despre angajați și departamentele în care lucrează.

1. Operatorul de compunere naturală în algebra relațională:

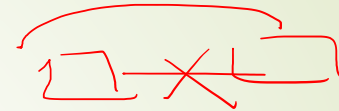
Rezultat = **JOIN** (SALARIAT, DEPARTAMENT).

2. Operatorul de compunere naturală în SQL:

```
SELECT  *  
FROM    salariat s, department d  
WHERE   s.cod_depart = d.cod_depart;
```

ALGEBRA RELAȚIONALĂ

θ-JOIN



- Operatorul θ -JOIN combină tupluri din două relații (nu neapărat corelate) cu condiția ca valorile atributelor specificate să satisfacă o anumită condiție specificată explicit în cadrul operației.
- Operatorul θ -JOIN este un operator derivat, fiind o combinație de produs scalar și selecție:

$$\text{JOIN}(R, S, \text{condiție}) = \sigma_{\text{condiție}}(R \times S)$$

Caz particular de θ -JOIN: equi-join (condiția utilizează operatorul "=")

JOB GRADERS.

JOBGR

ALGEBRA RELAȚIONALĂ

Exemplu. Să se afișeze pentru fiecare salariat, codul acestuia și grila sa de salarizare.

```
SELECT      empno, level
FROM        salgrade, emp
WHERE       sal BETWEEN losal AND hisal;
```

low = sal high = 2(?)

JOB GRADES

	<i>Grade</i>	<i>Low</i>	<i>High</i>	<i>Sup.</i>
1		600	2000	
2		2600	3700	
3		3800	4900	

Exemplu. Să se obțină informații despre contractanți (codul și banca) și obiectivele de investiție asociate acestora (denumire, număr certificat de urbanizare) cu condiția ca obiectivele să nu fie la aceeași adresă ca și contractanții.

1. Operatorul θ -JOIN în algebra relațională:

$R = \text{PROJECT}(\text{CONTRACTANT}, \text{cod_contractant}, \text{banca});$

$S = \text{PROJECT}(\text{OBIECTIV_INVESTITIE}, \text{denumire}, \text{nr_cert_urb});$

Rezultat = $\text{JOIN}(R, S, \text{OBIECTIV_INVESTITIE.adresa} \neq \text{CONTRACTANT.adresa}).$

2. Operatorul θ -JOIN în SQL:

```
SELECT      cod_contractant, banca, nr_cert_urb,
            denumire
FROM        contractant a, obiectiv_investitie b
WHERE       b.adresa <> a.adresa;
```

ALGEBRA RELAȚIONALĂ

SEMI-JOIN

- Operatorul SEMI-JOIN conservă **atributele unei singure relații participante** la compunere și este utilizat când nu sunt necesare toate atributele compunerii. Operatorul este asimetric.
- Tupluri ale relației R care participă în compunerea (naturală sau θ -JOIN) dintre relațiile R și S.
- SEMI-JOIN este un operator derivat, fiind o combinație de proiecție și compunere naturală sau proiecție și θ -JOIN:

$$\text{SEMIJOIN}(R, S) = \Pi_M(\text{JOIN}(R, S))$$

$$\text{SEMIJOIN}(R, S, \text{condiție}) = \Pi_M(\text{JOIN}(R, S, \text{condiție})),$$

unde am notat prin M atributele relației R .

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină informații referitoare la persoanele fizice (nume, carte_identitate) care investesc în obiective cu caracter recreativ.

1. Operatorul SEMI-JOIN în algebra relațională:

$R = \text{SELECT (OBJECTIV_INVESTITIE, denumire = 'cabana' OR denumire = 'casa de vacanta')}$

$S = \text{JOIN (PERS_FIZICA, R)}$

Rezultat = $\text{PROJECT (S, nume, carte_identitate)}$.

2. Operatorul SEMI-JOIN în SQL:

```
SELECT  nume, carte_identitate
FROM    pers_fizica a, obiectiv_investitie b
WHERE   a.cod_contractant = b.cod_contractant
AND     (denumire='cabana')
OR      (denumire= 'casa de vacanta');
```

*select nume, ci
from pers fiz.a
join obiectiv
on (a.cod contr =
b.cod contr)
where den =
cabana
sau
vacanta*

ALGEBRA RELAȚIONALĂ

OUTER JOIN

- Operația de **compunere externă** combină tupluri din două relații, tupluri pentru care nu sunt satisfăcute condițiile de corelare.
- În cazul aplicării operatorului JOIN se pot pierde tupluri, atunci când **există un tuplu în una din relații pentru care nu există nici un tuplu în cealaltă relație**, astfel încât să fie satisfăcută relația de corelare.
 - Operatorul elimină acest inconvenient prin **atribuirea valorii null valorilor atributelor care există într-un tuplu din una dintre relațiile de intrare, dar care nu există și în cea de-a doua relație.**
- **Practic, se realizează compunerea a două relații R și S la care se adaugă tupluri din R și S , care nu sunt conținute în compunere, completate cu valori *null* pentru attributele care lipsesc.**
- Compunerea externă poate fi: LEFT, RIGHT, FULL. De exemplu, R LEFT OUTER JOIN S reprezintă compunerea în care tuplurile din R , care nu au valori similare în coloanele comune cu relația S , sunt de asemenea incluse în relația rezultat.

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obțină informații referitoare la persoanele fizice care sunt investitori și la obiectivele lor de investiție industriale. Rezultatul va include și investitorii care nu au investit în obiective industriale.

$R = \text{SELECT (OBJECTIV_INVESTITIE, denumire = 'industrial')}$
 Rezultat = **LEFT OUTER JOIN** (PERS_FIZICA, R).

Select *
 From pers_fiz p left outer join
 ob_investitie a
 using(cod_contractant)
 Where denumire='ind';

Select *
 From pers_fiz p, ob_investitie a
 Where
 p.cod_contr=a.cod_contr(+)
 And denumire='ind';

Exemplu. Să se obțină informații referitoare la persoanele fizice care sunt investitori și la obiectivele lor de investiție industriale. Rezultatul va include obiectivele industriale care nu sunt construite de persoane fizice.

$R = \text{SELECT (OBJECTIV_INVESTITIE, denumire = 'industrial')}$
 Rezultat = **RIGHT OUTER JOIN** (PERS_FIZICA, R).

Select *
 From pers_fiz p right outer join
 ob_investitie a
 using(cod_contractant)
 Where denumire='ind';

Select *
 From pers_fiz p, ob_investitie a
 Where p.cod_contr (+)
 =a.cod_contr
 And denumire='ind';

Exemplu. Să se obțină informații referitoare la persoanele fizice care sunt investitori (chiar dacă nu au investit în obiective industriale) și la obiectivele lor de investiție industriale (chiar și cele care nu sunt construite de persoane fizice).

$R = \text{SELECT (OBJECTIV_INVESTITIE, denumire = 'industrial')}$
 Rezultat = **FULL OUTER JOIN** (PERS_FIZICA, R).

- Cum implementați în SQL?
- Operatorii algebrei relaționale pot fi reprezentați grafic cu ajutorul unor simboluri speciale. → curs!



TEMĂ

- Gestiunea unui lanț hotelier: diagrama conceptuală, scheme relaționale și exemple ale operatorilor algebrei relaționale: cerere în limbaj natural, expresie în algebra relațională și cerere în SQL.