BAZE DE DATE

CURS 6

ALGEBRA RELAȚIONALĂ

- **LDD** – precizează **entităţile**, relaţiile dintre ele, **atributele**, structura atributelor, **cheile**, **constrângerile**, prin urmare defineşte **structura obiectelor bazei de date** (**schema** bazei de date)*;*

- **LMD** – cuprinde aspecte referitoare la introducerea, modificarea, eliminarea şi căutarea datelor*;*

ALGEBRA RELAȚIONALĂ

- Modelul relaţional oferă **două mulţimi de operatori pe relaţii**:
  - **algebra relaţională** (filtrele se obţin aplicând operatori specializaţi asupra uneia sau mai multor relaţii din cadrul bazei relaţionale);
  - **calculul relaţional** (filtrele se obţin cu ajutorul unor formule logice pe care tuplurile rezultatului trebuie să le satisfacă);
  - Echivalenţa dintre algebra relaţională şi calculul relaţional a fost demonstrată de J.D.Ullman. Această echivalenţă arată că orice relaţie posibil de definit în algebra relaţională poate fi definită şi în cadrul calculului relaţional, şi reciproc.

ALGEBRA RELAȚIONALĂ

- **Algebra relaţională** a fost introdusă de E.F. Codd
  - mulţime de operaţii formale acţionând asupra unor relaţii şi având ca rezultat alte relaţii.

- Baza teoretică pentru limbajele de interogare relaţionale o constituie operatorii introduşi de Codd pentru prelucrarea relaţiilor.

ALGEBRA RELAȚIONALĂ

- Scopul fundamental al algebrei relaționale este de a permite scrierea **expresiilor relaționale**.
  - reprezentare de nivel superior, simbolică, a intenţiilor utilizatorului şi pot fi supuse unei diversităţi de reguli de transformare (**optimizare**).

- Relaţiile sunt închise faţă de algebra relaţională
  - operanzii şi rezultatele sunt relaţii → ieşirea unei operaţii poate deveni intrare pentru alta → posibilitatea imbricării expresiilor în algebra relaţională.

ALGEBRA RELAȚIONALĂ

- **Operatorii algebrei relaţionale** sunt:
  - operatori tradiţionali pe mulţimi (**UNION, INTERSECT, PRODUCT, DIFFERENCE**);
  - operatori relaţionali speciali (**PROJECT, SELECT, JOIN, DIVISION**).

ALGEBRA RELAȚIONALĂ

Operatorul PROJECT

- Proiecţia este o operaţie unară care elimină anumite atribute ale unei relaţii producând o submulţime „pe verticală“ a acesteia.
  - Suprimarea unor atribute poate avea ca efect apariţia unor tupluri duplicate, care trebuie eliminate.
- **Notaţii**:
  - $\Pi_{A1, \ldots, Am}$ **(*R*)**
  - **PROJECT (*R*, A_1, ..., A_m)**
  - $R[A_1, \ldots, A_m]$

unde A_1, A_2, ..., A_m sunt parametrii proiecţiei relativ la relaţia *R*.

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină o listă ce conţine numele, prenumele şi *job*-ul angajaţilor.

1. Proiecţie în algebra relaţională:

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

2. Proiecţie cu dubluri în SQL:

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

3. Proiecţie fără dubluri în SQL:

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

ALGEBRA RELAȚIONALĂ

Operatorul SELECT

- Selecţia (restricția) este o operaţie unară care produce o submulţime pe „orizontală“ a unei relaţii *R*.
  - Această submulţime se obţine prin extragerea tuplurilor din *R* care satisfac o condiţie specificată.
- **Notaţii**:
  - $\sigma_{\text{condiţie}}(R)$
  - *R*[condiţie]
  - **SELECT(*R*, condiţie)**
  - RESTRICT(*R*, condiţie).

ALGEBRA RELAȚIONALĂ

Exemplu. Să se obţină informaţii complete despre programatori.

1. Selecţie în algebra relaţională:

   Rezultat = **SELECT** (SALARIAT, job = ‘programator’)

2. Selecţie în SQL:

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

ALGEBRA RELAȚIONALĂ

Operatorul UNION

- Reuniunea a două relaţii *R* şi *S* este mulţimea tuplurilor aparţinând fie lui *R*, fie lui *S*, fie ambelor relaţii.
- **Notaţii**:
  - $R \cup S$
  - **UNION(*R*, *S*)**
  - OR(*R*, *S*)
  - APPEND(*R*, *S*).

***Exemplu*.** Să se obţină lista cu numele persoanelor fizice şi a subantreprenorilor.

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

ALGEBRA RELAȚIONALĂ

Operatorul DIFFERENCE

- Diferenţa a două relaţii *R* şi *S* este mulţimea tuplurilor care aparţin lui *R*, dar nu aparţin lui *S*.
- Diferenţa este o operaţie binară necomutativă care permite obţinerea tuplurilor ce apar numai într-o relaţie.
- **Notaţii**:
  - ***R – S***
  - **DIFFERENCE(*R*, *S*)**
  - REMOVE(*R*, *S*)
  - MINUS(*R*, *S*).

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină lista cu numărul contractului, tipul contractului, valoarea investiţiei şi durata lucrării pentru contractele de tip ‘T’ pentru care valoarea investiţiei nu depăşeşte 60000$.

1. Diferenţă în algebra relaţională:

   R = **PROJECT** (**SELECT** (CONTRACT, tip_contract=‘T’), nr_contract, tip_contract, val_investitie, durata_lucrare);

   S = **PROJECT** (**SELECT** (CONTRACT, val_investitie > 60000), nr_contract, tip_contract, val_investitie, durata_lucrare);

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

ALGEBRA RELAȚIONALĂ

2. Diferenţa 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 diferenţa se poate referi la tabele diferite!

ALGEBRA RELAȚIONALĂ

Operatorul INTERSECT

- Intersecţia a două relaţii *R* şi *S* este mulţimea tuplurilor care aparţin atât lui *R*, cât şi lui *S*. Operatorul INTERSECT este un operator binar, comutativ, derivat:
  - $R \cap S = R - (R - S)$
  - $R \cap S = S - (S - R)$.
- **Notaţii**:
  - **INTERSECT(*R*, *S*)**
  - ***R*** $\cap$ ***S***
  - AND(*R*, *S*).
- În anumite dialecte SQL există operator special (INTERSECT), care realizează această operaţie.
- Operatorii INTERSECT şi DIFFERENCE pot fi simulaţi în SQL (în cadrul comenzii SELECT) cu ajutorul opţiunilor EXISTS, NOT EXISTS, IN, NOT IN (!= ANY)

ALGEBRA RELAȚIONALĂ

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

1. Intersecţie în algebra relaţională:

   R = **PROJECT** (ANGAJATI, cod_sef);

   S = **PROJECT** (DEPARTAMENTE, cod_director),

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

2. Intersecţie în SQL:

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

ALGEBRA RELAȚIONALĂ

3. Simularea intersecţiei în SQL:

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

- Cum simulăm intersecția utilizând operatorul IN?
- Cum simulăm diferența utilizand NOT IN (Sa se obtina codurile departamentelor in care nu lucreaza nimeni)?

ALGEBRA RELAȚIONALĂ

Operatorul PRODUCT

- Fie *R* şi *S* relaţii de aritate ***m***, respectiv ***n***. Produsul cartezian al lui *R* cu *S* este mulţimea tuplurilor de aritate ***m + n*** unde primele *m* componente formează un tuplu în *R*, iar ultimele *n* componente formează un tuplu în *S*.
- **Notaţii:**
  - $R \times S$
  - **PRODUCT(*R*, *S*)**
  - TIMES(*R*, *S*).

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină lista tuturor posibilităţilor de investiţie în diverse obiective de către o firmă care este persoană juridică.

1. Produs cartezian în algebra relaţională:

   R = **PROJECT** (PERS_JURIDICA, nume, cod_contractant);

   S = **PROJECT** (OBIECTIV_INVESTITIE, denumire);

   Rezultat = **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 operaţie binară care defineşte o relaţie ce conţine valorile atributelor dintr-o relaţie care apar **în toate** valorile atributelor din cealaltă relaţie.
- **Notaţii**:
  - DIVIDE(*R*, *S*)
  - **DIVISION(*R*, *S*)**
  - ***R* ÷ *S*.**
- Diviziunea conţine acele tupluri de dimensiune $n - m$ la care, adăugând orice tuplu din *S*, se obţine un tuplu din *R*.
- Operatorul diviziune poate fi exprimat formal astfel:

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

ALGEBRA RELAȚIONALĂ

- Operatorul DIVISION este legat de cuantificatorul universal (∀) care nu există în SQL.

- Cuantificatorul universal poate fi însă simulat cu ajutorul cuantificatorului existenţial (∃) utilizând relaţia:

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

- 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 obţină codurile salariaţilor ataşaţi **tuturor** proiectelor pentru care s-a alocat un buget egal cu 1000.

1. Diviziune în algebra relaţională:

   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 funcţiei 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 informaţiei din mai multe relaţii corelate.
- Operatorul combină produsul cartezian, selecţia şi proiecţia.
- 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 relaţii *R* şi *S*, cu condiţia 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 defineşte o coloană în *R* şi o coloană în *S*, se selectează din $R \times S$ tuplurile ale căror valori coincid în coloanele ***R.A*** şi ***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:

$$\mathbf{JOIN(R, S) = \Pi_{i1,\ldots,im}\ \sigma_{(R.A1 = S.A1) \wedge \ldots \wedge (R.Ak = S.Ak)}(R \times S)},$$

unde $A_1, \ldots, A_k$ sunt atributele comune lui *R* şi *S*, iar $i_1, \ldots, i_m$ reprezintă lista componentelor din $R \times S$ (păstrând ordinea iniţială) din care au fost eliminate componentele $S.A_1, \ldots, S.A_k$.

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină informaţii complete despre angajaţi şi departamentele în care lucrează.

1. Operatorul de compunere naturală în algebra relaţională:

   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 relaţii (nu neapărat corelate) cu condiţia ca valorile atributelor specificate să satisfacă o anumită condiţie specificată explicit în cadrul operaţiei.
- Operatorul **θ**-JOIN este un operator derivat, fiind o combinaţie de produs scalar şi selecţie:

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

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

ALGEBRA RELAȚIONALĂ

Exemplu. Să se afişeze pentru fiecare salariat, codul acestuia şi grila sa de salarizare.

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

***Exemplu*.** Să se obţină informaţii despre contractanţi (codul şi banca) şi obiectivele de investiţie asociate acestora (denumire, număr certificat de urbanizare) cu condiţia ca obiectivele să nu fie la aceeaşi adresă ca şi contractanţii.

1. Operatorul **θ**-JOIN în algebra relaţională:

   R = **PROJECT** (CONTRACTANT, cod_contractant, banca);

   S = **PROJECT** (OBIECTIV_INVESTITIE, denumire, nr_cert_urb);

   Rezultat = **JOIN** (*R*, *S*, OBIECTIV_INVESTITIE.adresa <> 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 relaţii participante** la compunere şi este utilizat când nu sunt necesare toate atributele compunerii. Operatorul este asimetric.
  - Tupluri ale relaţiei R care participă în compunerea (naturală sau θ-JOIN) dintre relaţiile R şi S.

- SEMI-JOIN este un operator derivat, fiind o combinaţie de proiecţie şi compunere naturală sau proiecţie şi θ-JOIN:

  SEMIJOIN(*R*, *S*) = Π_M (JOIN(*R*, *S*))

  SEMIJOIN(*R*, *S*, condiţie) = Π_M (JOIN(*R*, *S*, condiţie)),

  unde am notat prin *M* atributele relaţiei *R*.

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină informaţii referitoare la persoanele fizice (nume, carte_identitate) care investesc în bunuri cu caracter recreativ (casa de vacanta sau cabana).

1. Operatorul SEMI-JOIN în algebra relaţională:

   R = **SELECT** (OBIECTIV_INVESTITIE, denumire = 'cabana' OR denumire = 'casa de vacanta')

   S = **JOIN** (PERS_FIZICA, *R*)

   Rezultat = **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');
```

ALGEBRA RELAȚIONALĂ

OUTER JOIN

- Operaţia de **compunere externă** combină tupluri din două relaţii, tupluri pentru care nu sunt satisfăcute condiţiile de corelare.
- În cazul aplicării operatorului JOIN se pot pierde tupluri, atunci când există un tuplu în una din relaţii pentru care nu există nici un tuplu în cealaltă relaţie, astfel încât să fie satisfăcută relaţia de corelare.
  - Operatorul elimină acest inconvenient prin atribuirea valorii *null* valorilor atributelor care există într-un tuplu din una dintre relaţiile de intrare, dar care nu există şi în cea de-a doua relaţie.
- **Practic, se realizează compunerea a două relaţii *R* şi *S* la care se adaugă tupluri din *R* şi *S*, care nu sunt conţinute în compunere, completate cu valori *null* pentru atributele 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 relaţia S, sunt de asemenea incluse în relaţia rezultat.

ALGEBRA RELAȚIONALĂ

***Exemplu*.** Să se obţină informaţii referitoare la persoanele fizice care sunt investitori şi la obiectivele lor de investiţie industriale. Rezultatul va include și investitorii care nu au investit în obiective industriale.

R = **SELECT** (OBIECTIV_INVESTITIE, denumire = 'industrial')

Rezultat = **LEFT OUTER JOIN** (PERS_FIZICA, *R*).

***Exemplu*.** Să se obţină informaţii referitoare la persoanele fizice care sunt investitori şi la obiectivele lor de investiţie industriale. Rezultatul va include obiectivele industriale care nu sunt construite de persoane fizice.

R = **SELECT** (OBIECTIV_INVESTITIE, denumire = 'industrial')

Rezultat = **RIGHT OUTER JOIN** (PERS_FIZICA, *R*).

***Exemplu*.** Să se obţină informaţii referitoare la persoanele fizice care sunt investitori (chiar dacă nu au investit în obiective industriale) şi la obiectivele lor de investiţie industriale (chiar şi cele care nu sunt construite de persoane fizice).

R = **SELECT** (OBIECTIV_INVESTITIE, denumire = 'industrial')

Rezultat = **FULL OUTER JOIN** (PERS_FIZICA, *R*).

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