Algebra Relațională

6

Limbaje de interogare relațională

- *Limbaj de interogare:* Permit manipularea şi regăsirea datelor dintr-o bază de date.
- Modelul relațional ofera suport pentru limbaje de interogare simple & puternice:
  - Fundament formal bazat pe logică.
  - Plajă largă de optimizări.
- Limbaje de interogare **!=** limbaje de programare!
  - nu sunt “Turing complete"
  - nu sunt utilizate pentru calcule complexe
  - oferă o modalitate simplă şi eficientă de acces la mulțimi de date voluminoase

Limbaje de interogare formale

- Două limbaje de interogare formează baza pentru limbajele utilizate în practică (ex. SQL):
  - *Algebra Relațională*: Mai operațională, utilă pentur reprezentarea planurilor de execuție.
  - *Relational Calculus*: Permite utilizatorilor să descrie **ce**, şi nu **cum** să obțină ceea ce doresc. (Non-operational, *declarativ*)

Algebra relațională

- O interogare se aplică *instanței* unei relații, şi rezultatul interogării reprezintă de asemenea o instanță de relație.
  - *Structura* relațiilor ce par într-o interogare este fixă (dar interogarea se va executa indiferent de instanța relației la un moemtn dat)
  - Structura *rezultatului* unei interogări este de asemenea fixă şi este determinată de definițiile construcțiilor limbajului de interogare.

- Notație pozițională sau prin nume:
  - Notația pozițională este mai utilă in definiții formale, însa utilizare numelor de câmpuri ori de câte ori se poate conduce la o interogare mai uşor de citit.
  - Ambele variante sunt utilizate în SQL

Algebra relațională

- Operatii de baza:
  - *Proiectia* (π) Elimina atributele nedorite ale unei relatii
  - *Selectie* (σ) Slecteaza o submultimes de tuple ale unei relatii.
  - *Prod cartezian* (X) Permite combinarea a doua relatii.
  - *Diferenta* (-) Tuplele ce sunt in in rel 1, dar nu in rel. 2.
  - *Reuniunea* (∪) Tuplele din rel. 1 si rel. 2.
- Operatii aditionale:
  - Intersectia, *join*, catul, redenumirea: nu sunt esentiale dar sunt foarte folositoare.
- Deoarece fiecare operatie returneaza o relatie, operatiile pot fi compuse (algebra este "inchisa".)

Proiecția

- L =(a_1, …, a_n) este o lista de atribute (sau *o lista de coloane*) ale relatiei R
- Se returneaza o relatie eliminand toate atributele care nu sunt in L

$$\pi_L(R) = \{ t \mid t_1 \in R \wedge$$
$$t.a_1 = t_1.a_1 \wedge$$
$$\ldots \wedge$$
$$t.a_n = t_1.a_n\}$$

Exemplu proiecție

$$\pi_{\text{cid, grade}}(\text{Enrolled})$$

$\pi_{\text{cid, grade}}$(

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7
1237	DB2	9`
1237	DB1	5
1237	Alg1	10

) =

cid	*grade*
Alg1	9
Alg1	10
DB1	10
DB2	9
DB1	7
DB1	5

Proiecția

Este $\pi_{cid,\ grade}$(Enrolled) echivalenta cu

```
SELECT cid, grade FROM Enrolled
```
?

Nu! Algebra relationala opereaza cu multimi => no exista duplicate.

```
SELECT DISTINCT cid, grade
FROM Enrolled
```

Selecția

- Selecteaza tuplele unei relatii R care verifica conditia *c* (numit si *predicat de selectie*).

$$\sigma_c(R) = \{ t \mid t \in R \wedge c \}$$

$$\sigma_{grade > 8}(Enrolled) = \{t \mid t \in Enrolled \wedge grade > 8 \}$$

$\sigma_{grade > 8}$(

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB2	9
1236	DB1	7
1237	DB1	5
1237	Alg1	6

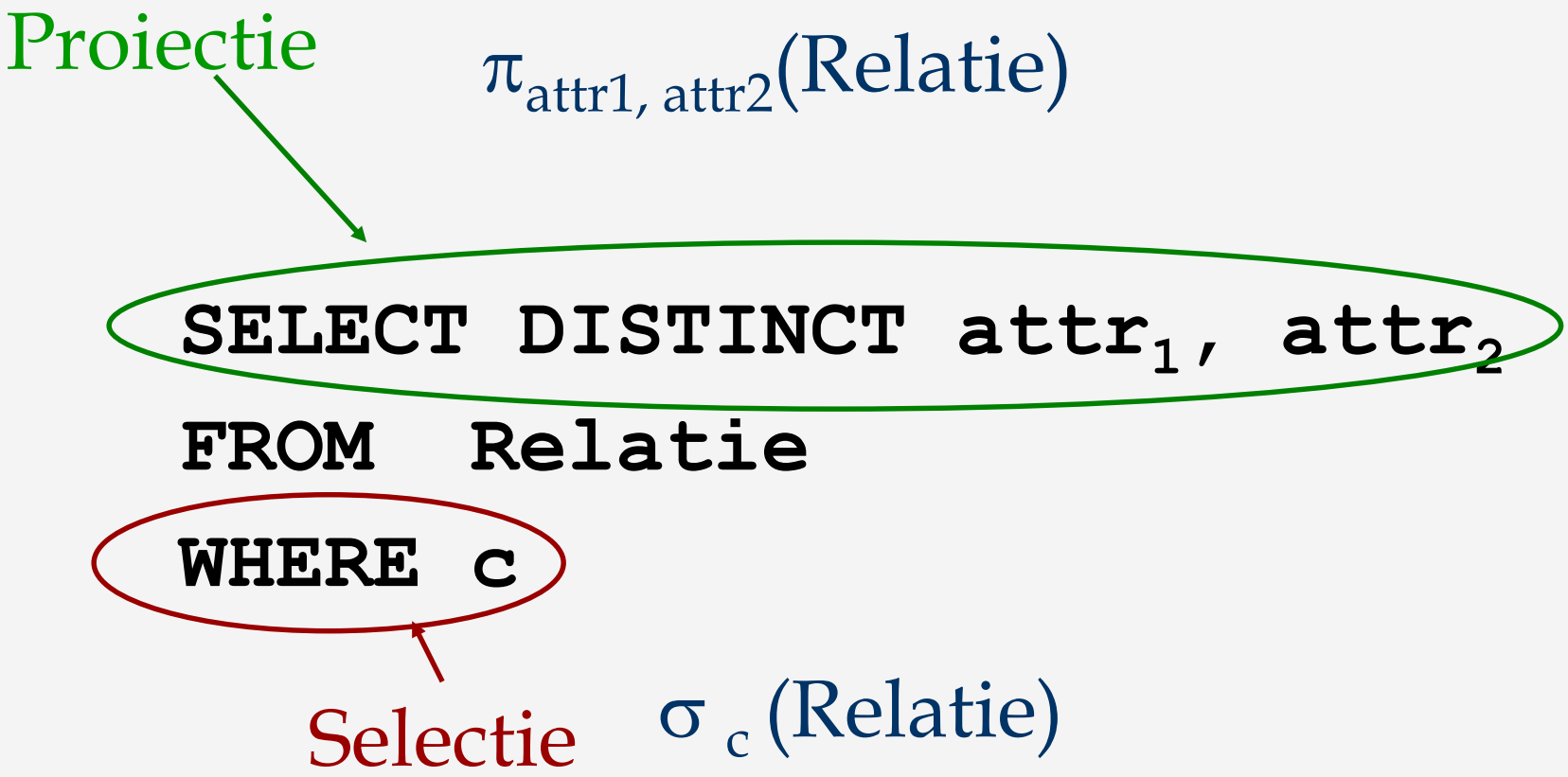
) =

sid	*cid*	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB2	9

Selecția

$$\sigma_{\text{grade} > 8}(\text{Enrolled})$$

```
SELECT DISTINCT *
FROM Enrolled
WHERE grade > 8
```


Proiectie
$\pi_{attr1, attr2}$(Relatie)
SELECT DISTINCT attr1, attr2
FROM Relatie
WHERE c
Selectie
σ_{c} (Relatie)

Condiția selecției

- **Term Op Term** este o condiție
  - unde **Term** este un nume de atribut
  - sau **Term** este o constanta
  - **Op** este un operator logic (ex. <, >, =, ≠ etc.)

- **(C1 ∧ C2)**, **(C1 ∨ C2)**, **(¬ C1)** sunt conditii formate din operatorii ∧ (si logic), ∨ (sau logic) sau ¬ (negatie), iar C1 si C2 sunt la randul lor conditii

Compunere

Rezultatul unei interogari este o relatie

$$\pi_{cid,\ grade}(\sigma_{\ grade > 8}(Enrolled))$$

$\pi_{cid,\ grade}(\sigma_{\ grade > 8}($

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7
1237	DB2	9`
1237	DB1	5
1237	Alg1	10

$)) =$

cid	*grade*
Alg1	9
Alg1	10
DB1	10
DB2	9

$$\pi_{\text{cid, grade}}(\sigma_{\text{grade} > 8}(\text{Enrolled}))$$

```
SELECT DISTINCT cid, grade
FROM  Enrolled
WHERE grade > 8
```

$$\sigma_{\text{grade} > 8}(\pi_{\text{cid, grade}}(\text{Enrolled}))$$

Care este interogarea SQL echivalenta?

Putem intotdeauna schimba ordinea operatorilor σ si π?

Reuniune, intersectie, diferenta

- $R_1 \cup R_2 = \{ t \mid t \in R_1 \vee t \in R_2 \}$
- $R_1 \cap R_2 = \{ t \mid t \in R_1 \wedge t \in R_2 \}$
- $R_1 - R_2 = \{ t \mid t \in R_1 \wedge t \notin R_2 \}$

Relatiile R_1 si R_2 trebuie sa fie *compatibile*:

- acelasi numar de atribute (aceeasi *aritate*)

- atributele aflate pe aceeasi pozitie au domenii *compatibile* si *acelasi nume*

Reuniune, intersectie, diferenta in SQL

$R_1 \cup R_2$

```
SELECT DISTINCT *
FROM R1
UNION
SELECT DISTINCT *
FROM R2
```

$R_1 \cap R_2$

```
SELECT DISTINCT *
FROM R1
INTERSECT
SELECT DISTINCT *
FROM R2
```

$R_1 - R_2$

```
SELECT DISTINCT *
FROM R1
EXCEPT
SELECT DISTINCT *
FROM R2
```

Sunt toti operatorii esentiali?

$$R_1 \cap R_2 = ((R_1 \cup R_2) - (R_1 - R_2)) - (R_2 - R_1)$$

Identifica tuplele
incluse in R_1 sau R_2

Elimina tuplele
ce apartin doar R_1

Elimina tuplele
ce apartin doar R_2

Produc cartezian

- Combinarea a doua relatii

$$R_1(a_1, \ldots, a_n) \text{ si } R_2(b_1, \ldots, b_m)$$

$$R_1 \, X \, R_2 = \{ t \mid t_1 \in R_1 \wedge t_2 \in R_2$$
$$\wedge \; t.a_1 = t_1.a_1 \ldots \wedge \; t.a_n = t_1.a_n$$
$$\wedge \; t.b_1 = t_2.b_1 \ldots \wedge \; t.b_m = t_2.b_m \}$$

```
SELECT DISTINCT *
FROM R1, R2
```

θ-Join

- Combinarea a doua relatii R_1 si R_2 cu respectarea conditiei *c*

$$R_1 \otimes_c R_2 = \sigma_c (R_1 \, X \, R_2)$$

$$\text{Students} \otimes_{\text{Students.sid=Enrolled.sid}} \text{Enrolled}$$

```
SELECT DISTINCT *
FROM Students,Enrolled
WHERE Students.sid =
Enrolled.sid
```

```
SELECT DISTINCT *
FROM Students
INNER JOIN Enrolled ON
Students.sid=Enrolled.sid
```

Equi-Join

- Combina doua relatii pe baza unei conditii compuse doar din egalitati ale unor atribute aflate in prima si a doua relatie si proiecteaza doar unul dintre atributele redundante (deoarece sunt egale)

$$R_1 \otimes_{E(c)} R_2$$

Courses

cid	*cname*
Alg1	Algorithms1
DB1	Databases1
DB2	Databases2

$\otimes_{E(\text{Courses.cid} = \text{Enrolled.cid})}$

Enrolled

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7

=

cname	*sid*	*cid*	*grad*
Algorithms1	1234	Alg1	9
Algorithms1	1235	Alg1	10
Databases1	1234	DB1	10
Databases2	1234	DB2	9
Databases1	1236	DB1	7

Join Natural

- Combina doua relatii pe baza egalitatii atributelor ce au *acelasi nume* si proiecteaza doar unul dintre atributele redundante

$$R_1 \otimes R_2$$

Courses

cid	*cname*
Alg1	Algorithms1
DB1	Databases1
DB2	Databases2

$\otimes$

Enrolled

sid	***cid***	*grade*
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7

=

cname	*sid*	*cid*	*grad*
Algorithms1	1234	Alg1	9
Algorithms1	1235	Alg1	10
Databases1	1234	DB1	10
Databases2	1234	DB2	9
Databases1	1236	DB1	7

Câtul

- Nu este un operator de baza, insa este util in anumite situatii
- Fie R_1 cu 2 atribute, x si y si R_2 cu un atribut y:

  $R_1 / R_2 = \{ <x> \mid \exists <x,y> \in R_1 \quad \forall <y> \in R_2 \}$

  adica, **R_1 / R_2 contine toate tuplele x a.î. pentru *toate* tuplele y din R_2, exista un tuplu xy in R_1.**

  Sau: Daca multimea valorilor y asociate cu o valoare x din R_1 contine toate valorile y din R_2, atunci x va fi returnat in rezultat R_1/R_2.
- Generalizând, x si y pot reprezentă orice multime de atribute; y este multimea atributelor din R_2, si $x \cup y$ reprezinta atributele lui R_1.

Modelarea operatorului *cât* folosind operatori de baza

- Cât-ul nu e un operator esential, ci doar "*scurtatura*".
  - (este si cazul operatorilor *join*, dar acestia sunt folositi mult mai des in interogari si au implementari speciale in diferite sisteme)
- *Ideea*: Pentru R_1 / R_2, vom determina valorile x care nu sunt `conenctate' cu anumite valori y din R_2.
  - valoarea x este *deconectata* daca atasand la ea o valoare y din R_2, obtinem un tuplu xy ce nu se regaseste in R_1.

Valorile x deconectate: $\pi_x ((\pi_x(R_1) \times R_2) - R_1)$

$R_1 / R_2 = \pi_x(R_1)$ - (*toate valorile deconectate*)

Redenumirea

- Daca atributele si relatiile au aceleasi nume (de exemplu la join-ul unei relatii cu ea insasi) estte necesar sa putem redenumi una din ele

$$\rho(R'\ (N_1 \rightarrow N'_1, N_2 \rightarrow N'_2\ ), R)$$

notatie alternativa: $\rho_{R'\ (N'1,\ N'2\ )}(R)$,

- Noua relatie R' are aceeasi instanta ca si R, iar structura sa contine atributul N'_i in locul atributului N_i

Redenumirea

$$\rho(\text{Courses2}\ (\text{cid} \rightarrow \text{code},\ \text{cname} \rightarrow \text{description}\ ),\ \text{Courses})$$

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

$\rightarrow$

Courses2

code	*description*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

```
SELECT cid as code,
       cname as description,
       credits
FROM Courses Courses2
```

Operatia de atribuire

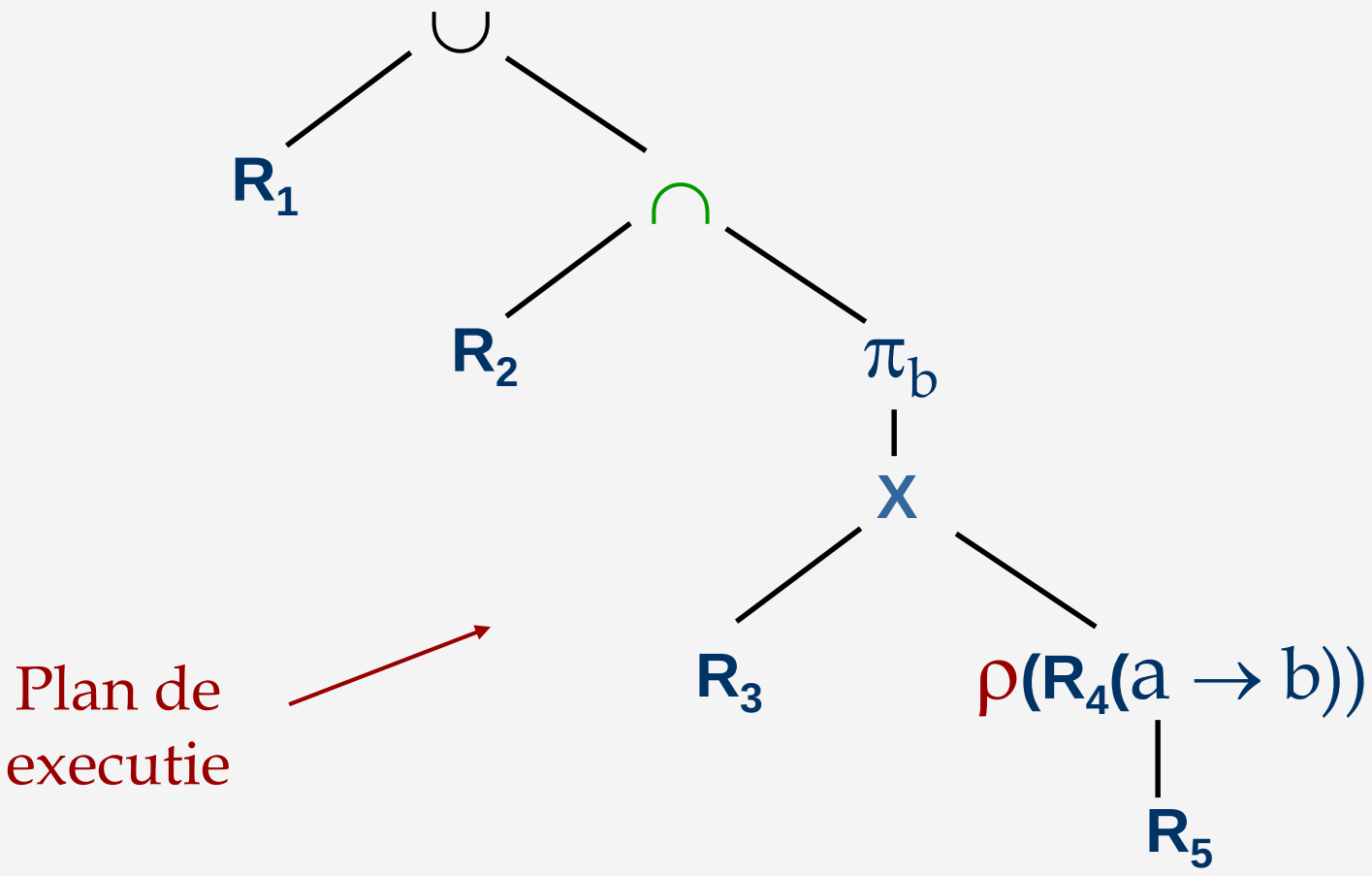
- Operatia de atribuire ($\leftarrow$) ofera un mod simplude tratare a interogarilor complexe.
  - Atribuirile se fac intotdeauna intr-o variabila temporara

$$Temp \leftarrow \pi_x(R_1 \; X \; R_2)$$

- Rezultatul expresiei din dreapta $\leftarrow$ este atribuit variabilei din stanga operatorului $\leftarrow$.
- Variabilele pot fi utilizate apoi in alte expresii
  - result $\leftarrow$ Temp – R_3

Expresii complexe

$$R_1 \cup (R_2 \cap \pi_b (R_3 \; X \; \rho(R_4(a \to b), R_5)))$$



Determinati numele tuturor studentilor cu note la cursul ‘BD1’

Solutie 1: π_{name} (($\sigma_{cid='BD1'}$(Enrolled)) $\otimes$ Students)

Solutie 2: ρ ($Temp_1$, $\sigma_{cid='BD1'}$(Enrolled))

ρ ($Temp_2$, $Temp_1 \otimes$ Students)

π_{name} ($Temp_2$)

Solutie 3: π_{name} ($\sigma_{cid='BD1'}$(Enrolled $\otimes$ Students))

Determinati numele tuturor studentilor cu note la cursuri cu 5 credite

- Informatia cu privire la credite se gaseste in in relatia *Courses*, si prin urmare se adauga un join natural:

$$\pi_{name}\ (\ (\sigma_{credits=5}(Courses)) \otimes Enrolled \otimes Students\ )$$

- O solutie mai eficienta:

$$\pi_{name}\ (\ \pi_{sid}(\pi_{cid}(\sigma_{credits=5}(Courses)) \otimes Enrolled) \otimes Students\ )$$

Modulul de optimizare a interogarilor e capabil sa transforme prima solutie in a doua!

Determinati numele tuturor studentilor cu note la cursuri cu 4 sau 5 credite

- Se identifica toate cursurile cu 4 sau 5 credite, apoi se determina studentii cu note la unul dintre aceste cursuri:

$$\rho\ (\mathit{TempCourses}, (\sigma_{\text{credits=4} \vee \text{credits=5}}(\text{Courses})))$$

$$\pi_{\text{name}}\ (\mathit{TempCourses} \otimes \text{Enrolled} \otimes \text{Students}\ )$$

- *TempCourses* se poate defini si utilizand reuniunea!

- Ce se intampla daca inlocuim $\vee$ cu $\wedge$ in interogare?

Determinati numele tuturor studentilor cu note la cursuri cu 4 si 5 credite

- Abordarea anterioara nu functioneaza! Trebuie identificati in paralel studentii cu note la cursuri de 4 credite si studentii cu note la cursuri de 5 credite, apoi se intersecteaza cele doua multimi (*sid* este cheie pentru *Students*):

$$\rho\ (Temp4, \pi_{sid}(\sigma_{credits=4}\ (\text{Courses}) \otimes \text{Enrolled}))$$

$$\rho\ (Temp5, \pi_{sid}(\sigma_{credits=5}\ (\text{Courses}) \otimes \text{Enrolled}))$$

$$\pi_{name}\ ((Temp4 \cap Temp5) \otimes \text{Students})$$

Determinati numele tuturor studentilor cu note la toate cursurile

- Se utilizeaza *câtul;* trebuie pregatite structurile relatiilor inainte de a folosi operatorul *cât*:

$$\rho\ (TempSIDs,\ \pi_{sid,\ cid}(\text{Enrolled}\ )\ /\ \pi_{cid}\ (\text{Courses}))$$

$$\pi_{name}(TempSIDs \otimes \text{Students})$$

Extensii ale operatorilor algebrici relationali

- Generalizarea proiectiei
- Functii de agregare
- Outer Join
- Modificarea bazei de date

Generalizarea proiectiei

- Operatorul *proiecție* este extins prin permiterea utilizarii functiilor aritmetice in lista de definire a proiectiei.

$$\pi_{F1,\, F2,\ldots,\, Fn}\,(R)$$

- R poate fi orice expresie din algebra relationala
- Fiecare dintr F_1, F_2, …, F_n sunt expresii aritmetice ce implica atribute din R si constante.

Functii de agregare

- **Functia de agregare** returneaza ca rezultat o valoare pe baza unei colectii de valori primite ca input

  avg: valoarea medie
  min: valoarea minima
  max: valoarea maxima
  sum: suma
  count: number of values

- **Operator de agregare** in algebra relationala

$$\mathcal{G}_{G_1, G_2, \ldots, G_n \quad F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(R)$$

- R poate fi orice expresie din algebra relationala
  - G_1, G_2 …, G_n e o lista de attribute pe baza carora se grupeaza datele (poate fi goala)
  - Fiecare F_i este o functie de agregare
  - Fiecare A_i este un nume de atribut

Aggregarea – Exemple

Relatie *R*:

A	*B*	*C*
α	α	7
α	β	7
β	β	3
β	β	10

$\mathcal{G}_{\text{sum(C)}}(R)$

sum(*C*)
27

- Rezultatul agregarii nu are un nume
  - se pot folosi operatorii de redenumire
  - se poate permite redenumirea ca parte a unei operatii de agregare

Outer Join

- Extensii ale operatorului join natural care impiedica pierderea informatiei:
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

- Realizeaza jonctiunea si apoi adauga la rezultat tuplele dintr-una din relatii (din stanga dreapta sau ambele parti ale operatorului) care nu sunt conectate cu tuple din celalta relatie.
- Utilizeaza valoarea *null*:
  - *null* semnifica faptul ca valoarea e necunoscuta sau nu exista
  - Toate comparatiile ce implica *null* sunt (simplu spus) **false** prin definitie.

Modificarea bazei de date

- Continutul bazei de date poate fi modificat folosind urmatorii operatori:
  - Stergere $R \leftarrow R - E$
  - Inserare $R \leftarrow R \cup E$
  - Modificare $R \leftarrow \pi_{F1, F2, \ldots, Fn}(R)$

- Toti acesti operatori sunt exprimati prin utilizarea operatorului de atribuire.