

Algebra Relațională

6

Limbaje de interogare relațională

- Limbaj de interogare: Permite manipularea și **regăsirea datelor** dintr-o bază de date.
- Modelul relațional oferă 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 și 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 **operatională**, utilă pentru reprezentarea planurilor de execuție.
 - Relational Calculus: Permite utilizatorilor să descrie **ce**, și nu **cum** să obțină ceea ce doresc. (Non-operational, declarativ)

Algebra relațională

- O interogare se aplică *instanței* unei relații, și rezultatul interogării reprezintă de asemenea o instanță de relație.
 - *Structura* relațiilor ce apar într-o interogare este fixă (dar interogarea se va executa indiferent de instanța relației la un moment dat)
 - Structura *rezultatului* unei interogări este de asemenea fixă și este determinată de definițiile construcțiilor limbajului de interogare.
- Notăție pozițională sau prin nume:
 - Notăția pozițională este mai utilă în definiții formale, însă utilizarea numelor de câmpuri conduce la interogări mai ușor de citit.
 - Ambele variante sunt utilizate în SQL

Algebra relațională

■ Operații de bază:

- Proiectia (π) Elimină attributele nedorite ale unei relații
- Selectie (σ) Selectează o submulțime de tupluri ale unei relații.
- Prod cartezian (\times) Permite combinarea a două relații.
- Diferenta ($-$) Tuplurile ce aparțin unei relații dar nu aparțin celeilalte
- Reuniunea (\cup) Tuplurile aparținând ambelor relații

■ Operații adiționale:

- Intersecția, join, câțul, redenumirea: nu sunt esențiale dar sunt foarte folositoare.
- Deoarece fiecare operație returnează o relație, **operațiile pot fi compuse** (algebra este “închisă”).)

Proiecția

- $L = (a_1, \dots, a_n)$ este o listă de attribute (sau *o lista de coloane*) ale relației R
- Returnează o relație eliminând toate attributele care nu sunt în L

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

Exemplu proiecție

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

$\pi_{cid, grade}(\$

| <i>sid</i> | <i>cid</i> | <i>grade</i> |
|------------|------------|--------------|
| 1234 | Alg1 | 9 |
| 1235 | Alg1 | 10 |
| 1234 | DB1 | 10 |
| 1234 | DB2 | 9 |
| 1236 | DB1 | 7 |
| 1237 | DB2 | 9 |
| 1237 | DB1 | 5 |
| 1237 | Alg1 | 10 |

) =

| <i>cid</i> | <i>grade</i> |
|------------|--------------|
| Alg1 | 9 |
| Alg1 | 10 |
| DB1 | 10 |
| DB2 | 9 |
| DB1 | 7 |
| DB1 | 5 |

Proiecția

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

SELECT cid, grade FROM Enrolled ?

Nu! Algebra relațională operează cu mulțimi
=> nu există duplicate.

SELECT DISTINCT cid, grade
FROM Enrolled

Selecția

- Selectează tuplurile unei relații **R** care verifică o condiție **c** (numită și *predicat de selecție*).

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

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

| <i>sid</i> | <i>cid</i> | <i>grade</i> |
|------------|------------|--------------|
| 1234 | Alg1 | 9 |
| 1235 | Alg1 | 10 |
| 1234 | DB2 | 9 |
| 1236 | DB1 | 7 |
| 1237 | DB1 | 5 |
| 1237 | Alg1 | 6 |

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

| <i>sid</i> | <i>cid</i> | <i>grade</i> |
|------------|------------|--------------|
| 1234 | Alg1 | 9 |
| 1235 | Alg1 | 10 |
| 1234 | DB2 | 9 |

$) =$

Selecția

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

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

Proiecție

$\pi_{\text{attr1}, \text{attr2}}(\text{Relație})$



SELECT DISTINCT attr₁, attr₂
FROM Relație



WHERE c

Selecție $\sigma_c(\text{Relație})$

Condiția selecției

- **Term Op Term** este o condiție, unde
 - **Term** este un nume de atribut, sau
 - **Term** este o constantă
 - **Op** este un operator logic (ex. $<$, $>$, $=$, \neq etc.)
- **$(C1 \wedge C2)$, $(C1 \vee C2)$, $(\neg C1)$** sunt condiții formate din operatorii \wedge (și logic), \vee (sau logic) sau \neg (negație), iar $C1$ și $C2$ sunt la rândul lor condiții

Compunere

Rezultatul unei interogări este o relație

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

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

| <i>sid</i> | <i>cid</i> | <i>grade</i> |
|------------|------------|--------------|
| 1234 | Alg1 | 9 |
| 1235 | Alg1 | 10 |
| 1234 | DB1 | 10 |
| 1234 | DB2 | 9 |
| 1236 | DB1 | 7 |
| 1237 | DB2 | 9 |
| 1237 | DB1 | 5 |
| 1237 | Alg1 | 10 |

| <i>cid</i> | <i>grade</i> |
|------------|--------------|
| 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 echivalentă?

Putem schimba întotdeauna ordinea operatorilor σ și π ?

Reuniune, intersecție, diferență

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

Relațiile R_1 și R_2 trebuie să fie *compatibile*:

- același număr de atribute (aceeași *aritate*)
- atributele aflate pe aceeași poziție au domenii *compatibile* și *același nume*

Reuniune, intersecție, diferență în 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
```


Toți sunt operatorii esențiali?

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

Identifică tuplurile
incluse în R_1 sau R_2



Elimină tuplurile
ce aparțin doar R_1



Elimină tuplurile
ce aparțin doar R_2



Produs cartezian

- Combinarea a doua relații

$$R_1(a_1, \dots, a_n) \text{ și } R_2(b_1, \dots, b_m)$$

$$R_1 \times R_2 = \{ t \mid t_1 \in R_1 \wedge t_2 \in R_2 \\ \wedge t.a_1 = t_1.a_1 \dots \wedge t.a_n = t_1.a_n \\ \wedge t.b_1 = t_2.b_1 \dots \wedge t.b_m = t_2.b_m \}$$

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

θ -Join

- Combinarea a doua relații R_1 și R_2 cu respectarea condiției c

$$R_1 \otimes_c R_2 = \sigma_c (R_1 \times R_2)$$

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

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

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

Equi-Join

- Combină două relații pe baza unei condiții compuse doar din egalități ale unor attribute aflate în prima și a doua relație și proiectează doar unul dintre attributele redundante (deoarece sunt egale)

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

Courses

| <i>cid</i> | <i>cname</i> |
|------------|--------------|
| Alg1 | Algorithms1 |
| DB1 | Databases1 |
| DB2 | Databases2 |

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

Enrolled

| <i>sid</i> | <i>cid</i> | <i>grade</i> |
|------------|------------|--------------|
| 1234 | Alg1 | 9 |
| 1235 | Alg1 | 10 |
| 1234 | DB1 | 10 |
| 1234 | DB2 | 9 |
| 1236 | DB1 | 7 |

=

| <i>cname</i> | <i>sid</i> | <i>cid</i> | <i>grad</i> |
|--------------|------------|------------|-------------|
| Algorithms1 | 1234 | Alg1 | 9 |
| Algorithms1 | 1235 | Alg1 | 10 |
| Databases1 | 1234 | DB1 | 10 |
| Databases2 | 1234 | DB2 | 9 |
| Databases1 | 1236 | DB1 | 7 |

Join Natural

- Combină două relații pe baza egalității atributelor ce au *același nume* și proiectează doar unul dintre attributele redundante

$$R_1 \otimes R_2$$

| <i>Courses</i> | | | <i>Enrolled</i> | | | | |
|----------------|--------------|---|-----------------|--------------|--------------|------------|-------------|
| <i>cid</i> | <i>cname</i> | | <i>sid</i> | <i>cid</i> | <i>grade</i> | | |
| Alg1 | Algorithms1 | ⊗ | 1234 | Alg1 | 9 | | |
| DB1 | Databases1 | | 1235 | Alg1 | 10 | | |
| DB2 | Databases2 | | 1234 | DB1 | 10 | | |
| | | | 1234 | DB2 | 9 | | |
| | | | 1236 | DB1 | 7 | | |
| | | | = | | | | |
| | | | | <i>cname</i> | <i>sid</i> | <i>cid</i> | <i>grad</i> |
| | | | | 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 bază, însă este util în anumite situații (simplifică mult interogarea)

- Fie R_1 cu 2 attribute, x și y și R_2 cu un atribut y :

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

adică, **R_1 / R_2 conține toate tuplurile x a.î. pentru fiecare dintre tuplurile y din R_2 , există câte un tuplu xy în R_1 .**

Sau: Dacă mulțimea valorilor y asociate cu o valoare x din R_1 conține toate valorile y din R_2 , atunci x va fi returnat în rezultat R_1 / R_2 .

- Generalizând, x și y pot reprezenta orice multime de attribute; y este mulțimea atributelor din R_2 , și $x \cup y$ reprezintă attributele lui R_1 .

Modelarea operatorului *cât* folosind operatori de bază

- Cât-ul nu e un operator esențial, ci doar o "scurtătură".
 - (este și cazul operatorilor *join*, dar aceștia sunt folosiți mult mai des în interogări și au implementări speciale în 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 *deconectată* dacă atașând la ea o valoare y din R_2 , obținem un tuplu xy ce nu se regăsește în R_1 .

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

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

Redenumirea

- Dacă attributele și relațiile au aceleași nume (de exemplu la *join*-ul unei relații cu ea însăși) este necesar să putem redenumi una din ele

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

notație alternativă: $\rho_{R' (N'_1, N'_2)}(R)$,

- Noua relație R' are aceeași instanță ca R , iar structura sa conține atributul N'_i în locul atributului N_i

Redenumirea

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

Courses

| <i>cid</i> | <i>cname</i> | <i>credits</i> |
|------------|--------------|----------------|
| Alg1 | Algorithms1 | 7 |
| DB1 | Databases1 | 6 |
| DB2 | Databases2 | 6 |



Courses2

| <i>code</i> | <i>description</i> | <i>credits</i> |
|-------------|--------------------|----------------|
| Alg1 | Algorithms1 | 7 |
| DB1 | Databases1 | 6 |
| DB2 | Databases2 | 6 |

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

Operația de atribuire

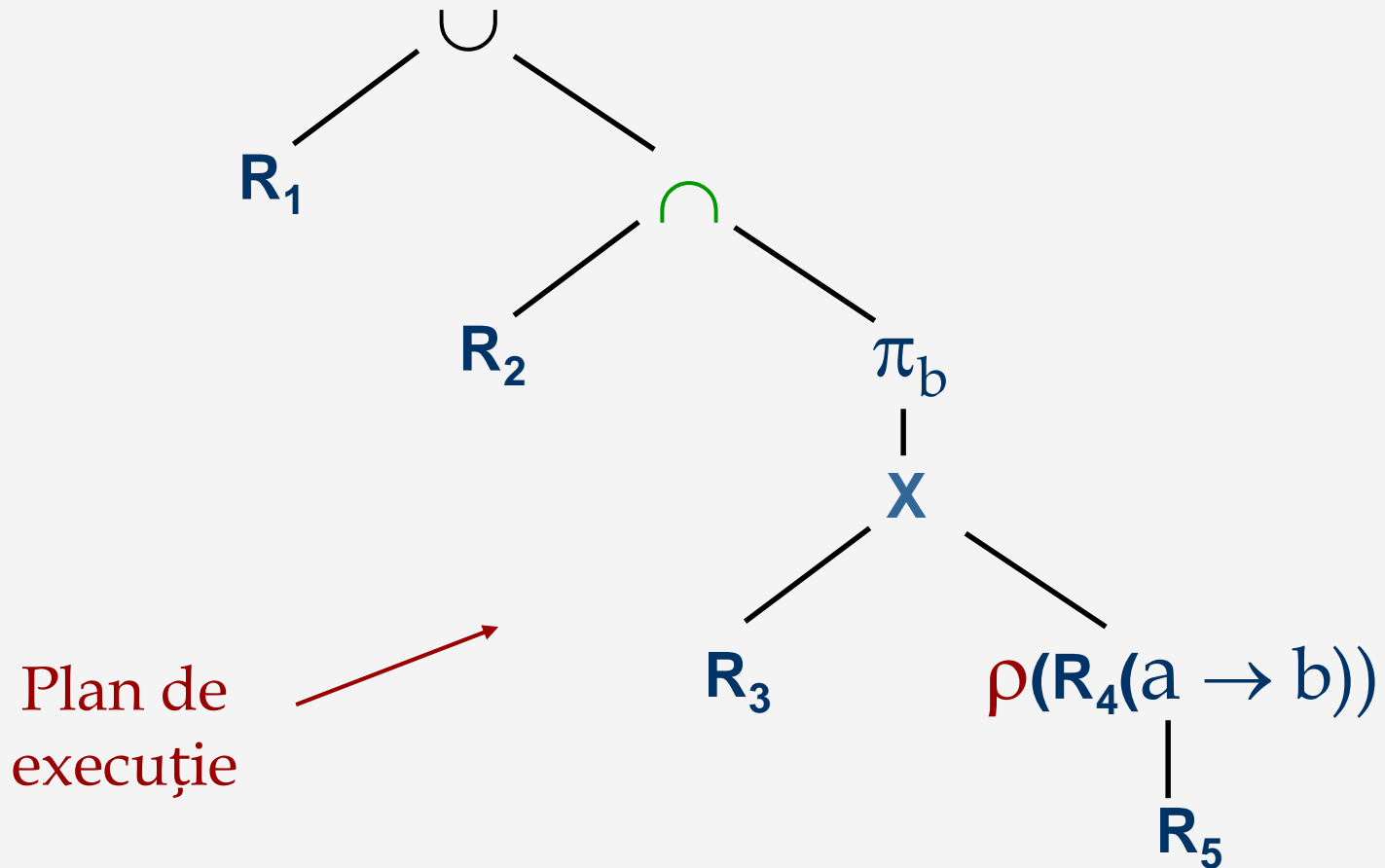
- Operația de atribuire (\leftarrow) oferă un mod simplu de tratare a interogărilor complexe.
 - Atribuirile se fac întotdeauna într-o variabilă temporară

$$\text{Temp} \leftarrow \pi_x(R_1 \times R_2)$$

- Rezultatul expresiei din dreapta \leftarrow este atribuit variabilei din stânga operatorului \leftarrow .
- Variabilele pot fi utilizate apoi în alte expresii
 - $\text{result} \leftarrow \text{Temp} - R_3$

Expresii complexe

$$R_1 \cup (R_2 \cap \pi_b (R_3 \bowtie \rho(R_4(a \rightarrow b), R_5)))$$



Determinați numele tuturor studenților cu note la cursul 'BD1'

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

Solutie 2: $\rho (\text{Temp}_1, \sigma_{\text{cid}='BD1'}(\text{Enrolled}))$
 $\rho (\text{Temp}_2, \text{Temp}_1 \otimes \text{Students})$
 $\pi_{\text{name}} (\text{Temp}_2)$

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

Determinați numele tuturor studenților cu note la cursuri cu 5 credite

- Informația cu privire la credite se găsește în relația *Courses*, și prin urmare se adaugă un join natural:

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

- O soluție mai eficientă:

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

Modulul de optimizare a interogărilor e capabil să transforme prima soluție în a doua!

Determinați numele tuturor studenților cu note la cursuri cu 4 sau 5 credite

- Se identifică toate cursurile cu 4 sau 5 credite, apoi se determină studenții cu note la unul dintre aceste cursuri:

$$\rho (TempCourses, (\sigma_{credits=4 \vee credits=5}(Courses)))$$
$$\pi_{name} (TempCourses \otimes Enrolled \otimes Students)$$

- *TempCourses* se poate defini și utilizând reuniunea!
- Ce se întâmplă dacă înlocuim \vee cu \wedge în interogare?

Determinați numele tuturor studenților cu note la cursuri cu 4 și 5 credite

- Abordarea anterioară nu funcționează! Trebuie identificați în paralel studenții cu note la cursuri de 4 credite și studenții cu note la cursuri de 5 credite, apoi se intersectează cele două mulțimi (*sid* este cheie pentru *Students*):

$$\rho (Temp4, \pi_{sid}(\sigma_{credits=4} (Courses) \otimes Enrolled))$$
$$\rho (Temp5, \pi_{sid}(\sigma_{credits=5} (Courses) \otimes Enrolled))$$
$$\pi_{name} ((Temp4 \cap Temp5) \otimes Students)$$

Determinați numele tuturor studenților cu note la toate cursurile

- Se utilizează *câtul*; trebuie pregătite structurile relațiilor înainte de a folosi operatorul *cât*:

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

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

Extensii ale operatorilor algebrici relaționali

- Generalizarea proiecției
- Funcții de agregare
- Outer Join
- Modificarea bazei de date

Generalizarea proiecției

- Operatorul *proiecție* este extins prin permiterea utilizării funcțiilor aritmetice în lista de definire a proiecției.

$$\pi_{F_1, F_2, \dots, F_n}(R)$$

- R poate fi orice expresie din algebra relatională
- Fiecare dintre F_1, F_2, \dots, F_n sunt expresii aritmetice ce implică attribute din R și constante.

Funcții de agregare

- **Funcția de agregare** returneaza ca rezultat o valoare pe baza unei colecții de valori primite ca input

avg: valoarea medie

min: valoarea minimă

max: valoarea maximă

sum: suma

count: numărul înregistrărilor

- **Operator de agregare în algebra relațională**

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

- R poate fi orice expresie din algebra relațională
 - G_1, G_2, \dots, G_n e o listă de attribute pe baza cărora se grupează datele (poate fi goală)
 - Fiecare F_i este o funcție de agregare
 - Fiecare A_i este un nume de atribut

Aggregarea – Example

Relatie
 R :

| A | B | C |
|----------|----------|-----|
| α | α | 7 |
| α | β | 7 |
| β | β | 3 |
| β | β | 10 |

$g_{\text{sum}(C)}(R)$

| 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 împiedică pierderea informației:

- Left Outer Join



- Right Outer Join



- Full Outer Join



■ Realizează joncțiunea și apoi adaugă la rezultat tuplurile dintr-una din relații (din stânga, dreapta sau ambele părți ale operatorului) care nu sunt conectate cu tupluri din celaltă relație.

■ Utilizează valoarea *null*:

- *null* semnifică faptul că valoarea e necunoscută sau nu există

- Toate comparațiile ce implică *null* sunt (*simply spus*) **false** prin definiție.

Modificarea bazei de date

■ Conținutul bazei de date poate fi modificat folosind următorii operatori:

■ Ștergere $R \leftarrow R - E$

■ Inserare $R \leftarrow R \cup E$

■ Modificare $R \leftarrow \pi_{F_1, F_2, \dots, F_n}(R)$

■ Toți acești operatori sunt exprimați prin utilizarea operatorului de atribuire.