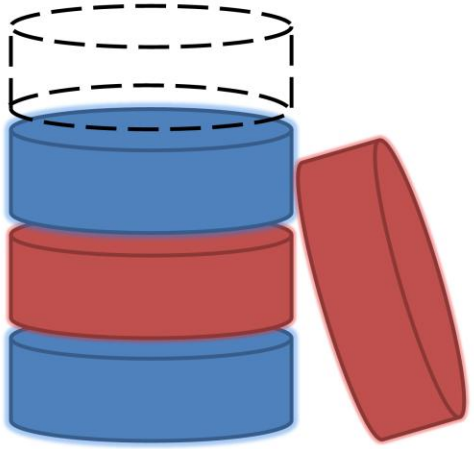
Algebra Relațională



6

Limbaje de interogare relațională

- *Limbaj de interogare:* Permite manipularea şi 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 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 operatională, utilă pentru 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 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 şi este determinată de definițiile construcțiilor limbajului de interogare.

- Notație pozițională sau prin nume:
  - Notația pozițională este mai utilă în definiții formale, însă utilizarea numelor de câmpuri conduce la interogări mai uşor de citit.
  - Ambele variante sunt utilizate în SQL

Algebra relațională

- Operatori de bază:
  - *Proiectia* (π) Elimină atributele nedorite ale unei relații
  - *Selectie* (σ) Selectează o submulțime de tupluri ale unei relații.
  - *Prod cartezian* (X) Permite combinarea a două relații.
  - *Diferenta* (-) Tuplurile ce aparțin unei relații dar nu aparțin celeilalte
  - *Reuniunea* (∪) Tuplurile aparținând ambelor relații
- Operatori adiționali:
  - Intersecția, *join*, câtul, redenumirea: nu sunt esențiale dar sunt foarte folositoare.
- Deoarece fiecare operator returnează o relație, operatorii pot fi compuși (algebra este "închisă".)

Proiecția

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

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

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

$\sigma_{grade > 8}($

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
```

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. <, >, =, ≠ etc.)

- **(C1 ∧ C2)**, **(C1 ∨ C2)**, **(¬ C1)** sunt condiții formate din operatorii ∧ (*și* logic), ∨ (*sau* logic) sau ¬ (*negație*), iar C1 și C2 sunt la rândul lor condiții

Compunere

Rezultatul unei interogări este o relație

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

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

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

Proiecție

$\pi_{attr1,\ attr2}$(Relație)

```
SELECT DISTINCT attr1, attr2
FROM  Relatie
WHERE c
```

Selecție

σ_{c} (Relație)

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

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

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

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

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

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

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

θ-Join

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

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

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

```
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 atribute aflate în prima și a doua relație și proiectează 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*	*grade*
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 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*	*grade*
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 atribute, x și y și R_2 cu un atribut y:

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

  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 atribute; y este mulțimea atributelor din R_2, și $x \cup y$ reprezintă atributele 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 in diferite sisteme)
- *Ideea*: Pentru R_1 / R_2, vom determina valorile x care nu sunt `conectate' 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: $\pi_x ((\pi_x(R_1) \text{ X } R_2) - R_1)$

$$R_1 / R_2 = \pi_x(R_1) - (\textit{Valorile x deconectate})$$

Redenumirea

- Dacă atributele 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

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
```

Operatorul de atribuire

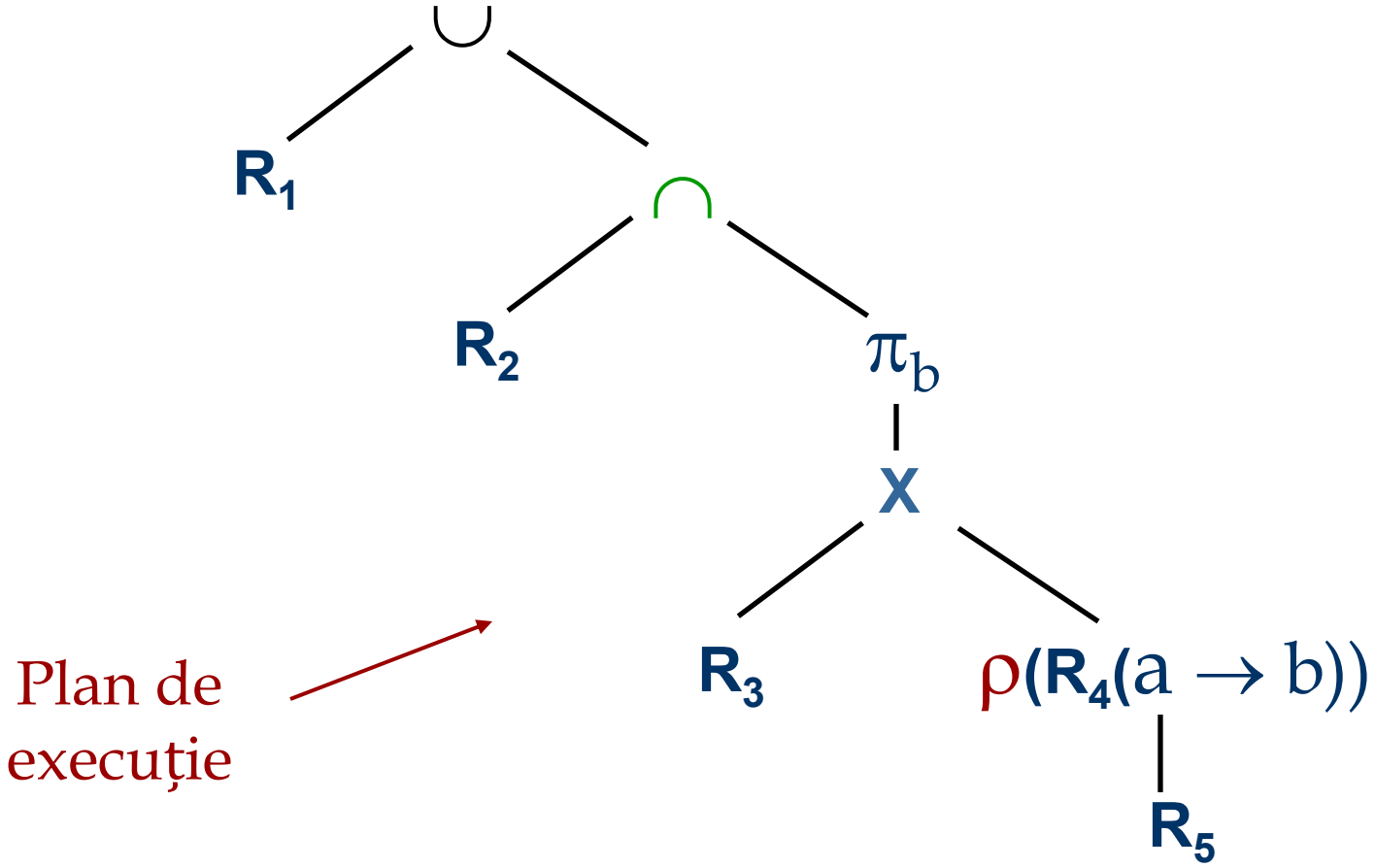
- Operatorul de atribuire ($\leftarrow$) oferă un mod simplu de tratare a interogărilor complexe.
  - Atribuirile se fac intotdeauna într-o variabilă temporară

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

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

Expresii complexe

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



Determinați numele tuturor studenților 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))

Determinați numele tuturor studenților cu note la cursuri cu 5 credite

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

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

- O soluție mai eficientă:

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

Modulul de optimizare a interogărilor e capabil să transforme prima solutie 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:

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

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

- *TempCourses* se poate defini și utilizând reuniunea!

- Ce se întâmplă dacă înlocuim $\vee$ cu $\wedge$ în interogare?

Determinați numele tuturor studenților cu note la cursuri cu 4 <u>si</u> 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*:

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

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