

# BAZE DE DATE

Procesarea interogărilor

@FII (2011-2012)

prezentat de Mihaela Elena Breabăn

# Tematică curs

---

- ▶ **Proiectarea bazelor de date relaționale**

- ▶ Normalizare și denormalizare
- ▶ Modelul entitate-asociere, diagrame UML
- ▶ Constrângeri și declanșatoare
- ▶ View-uri
- ▶ Indecși

- ▶ **Procesarea interogărilor**

- ▶ Managementul tranzacțiilor
- ▶ OLAP, Baze de date distribuite, NoSQL, Data Mining

# Cuprins

---

- ▶ Etapele procesării interogărilor
- ▶ Expresii în algebra relațională
  - ▶ Operatori (revizitat)
  - ▶ Expresii
  - ▶ Echivalența expresiilor
- ▶ Estimarea costului interogării
- ▶ Algoritmi pentru evaluarea operatorilor/expresiilor în algebra relațională

# Etapele procesării interogărilor

## ► Compilarea interogării

### ► Analiza sintactică

#### ► Parsare

- Arbore de parsare

### ► Analiza semantică

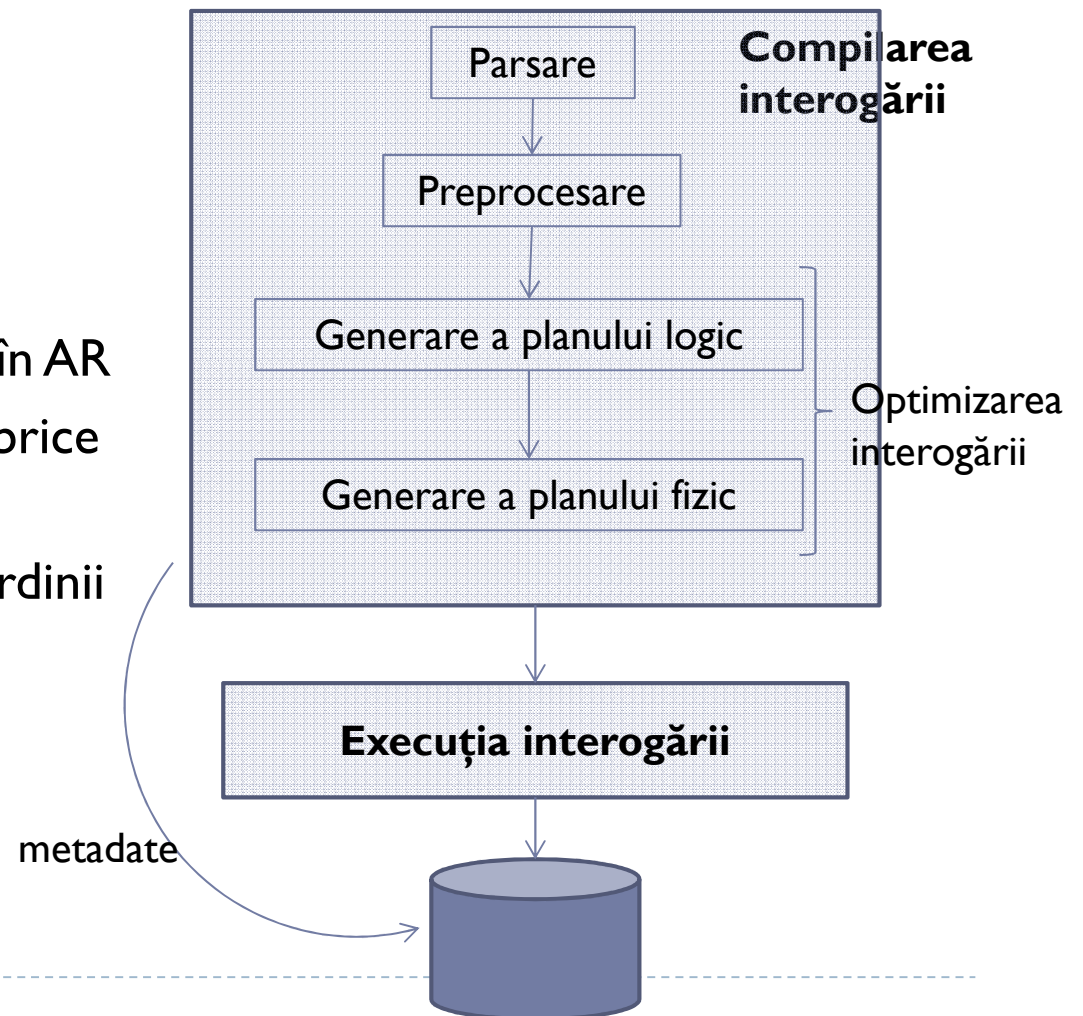
#### ► Preprocesare și rescriere în AR

#### ► Selecția reprezentării algebrice

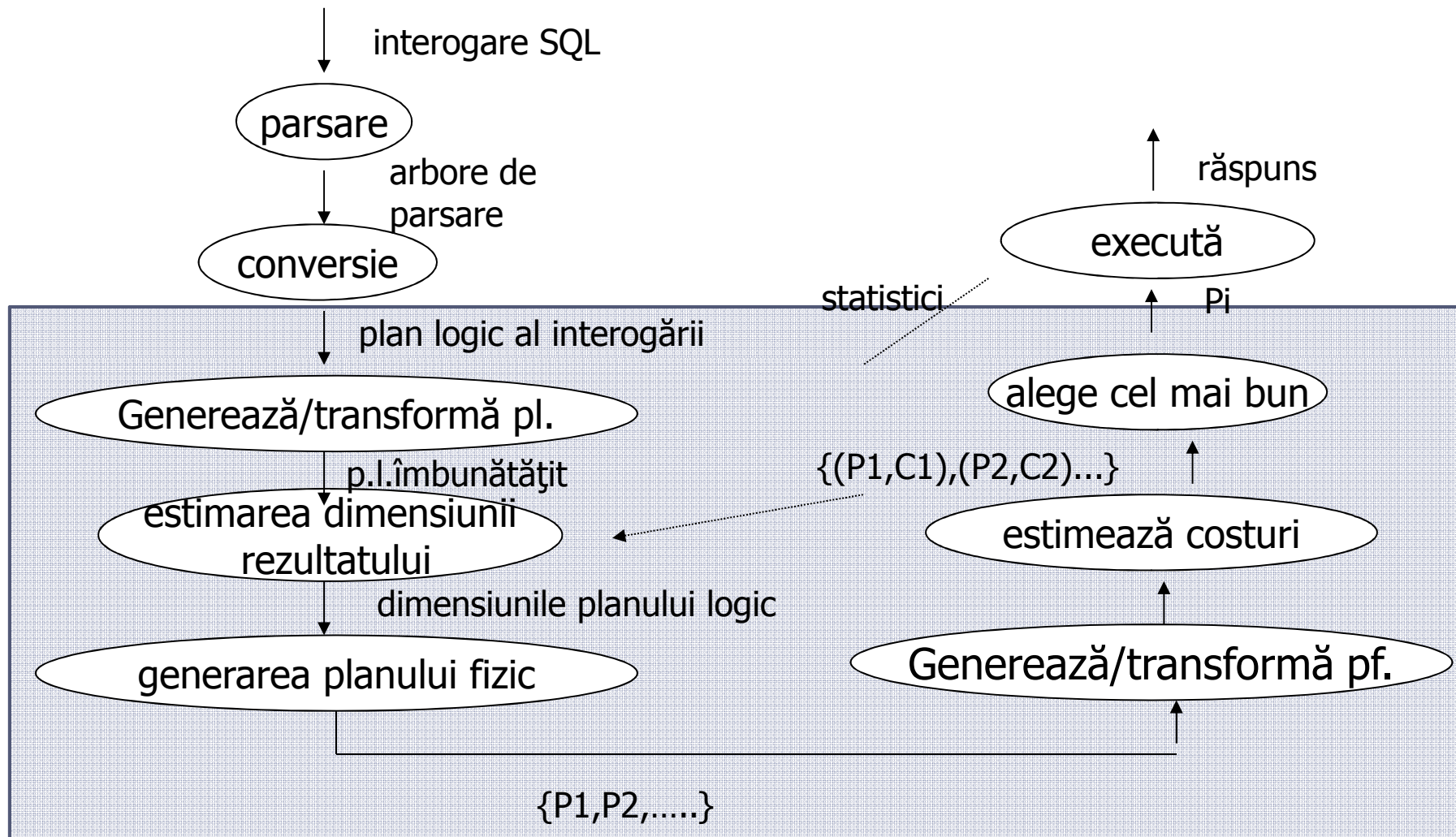
- Plan logic

#### ► Selecția algoritmilor și a ordinii

- Plan fizic

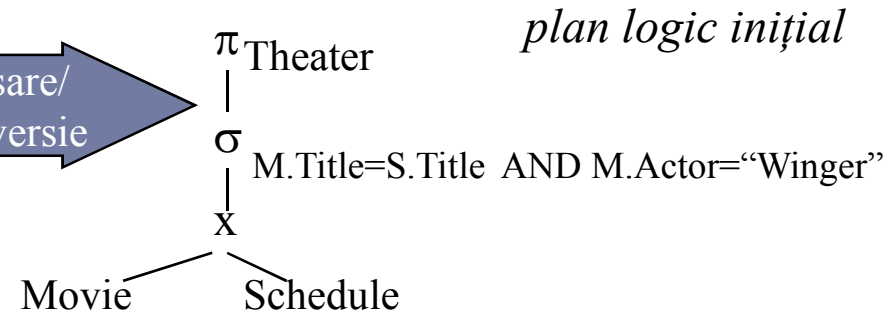


# Optimizarea interogărilor

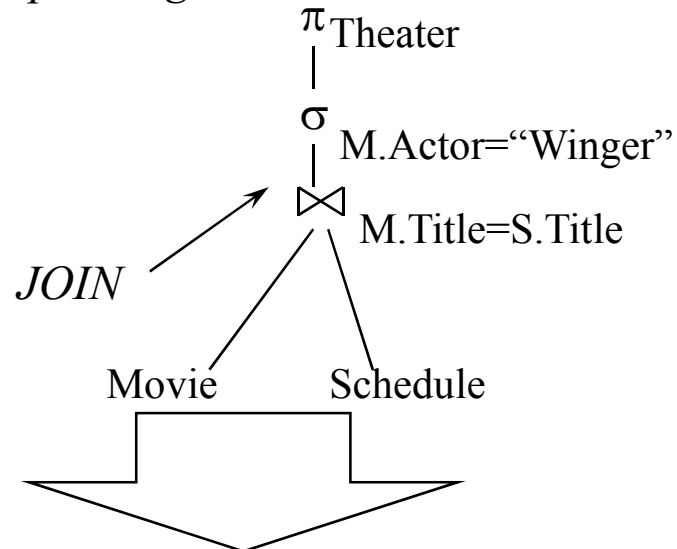


# Drumul unei interogări

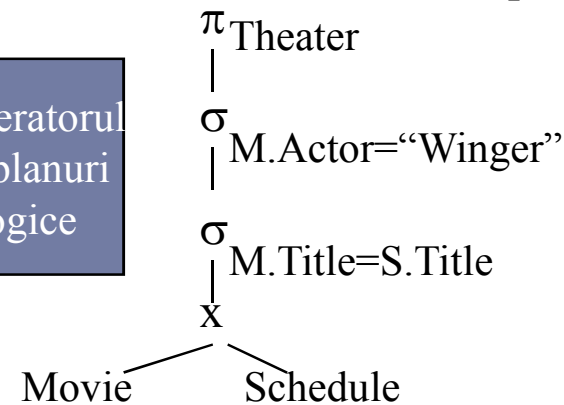
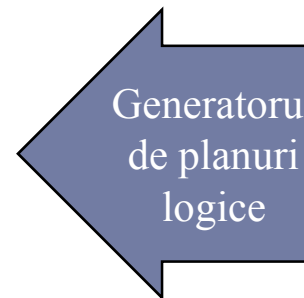
SELECT Theater  
FROM Movie M, Schedule S  
WHERE  
M.Title = S.Title  
AND M.Actor="Winger"



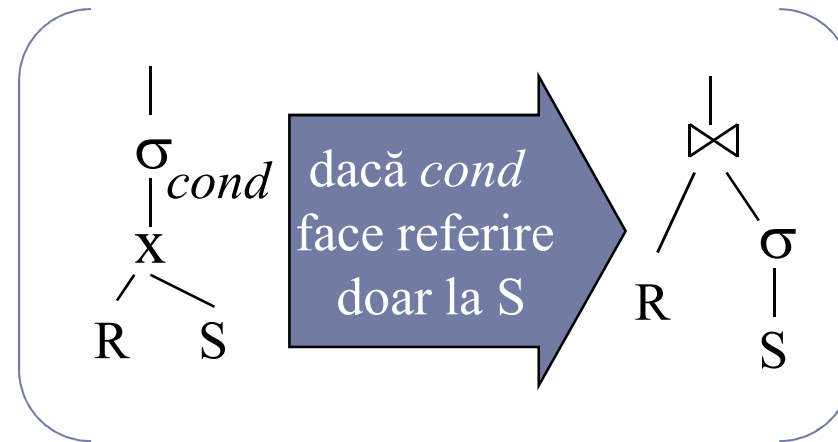
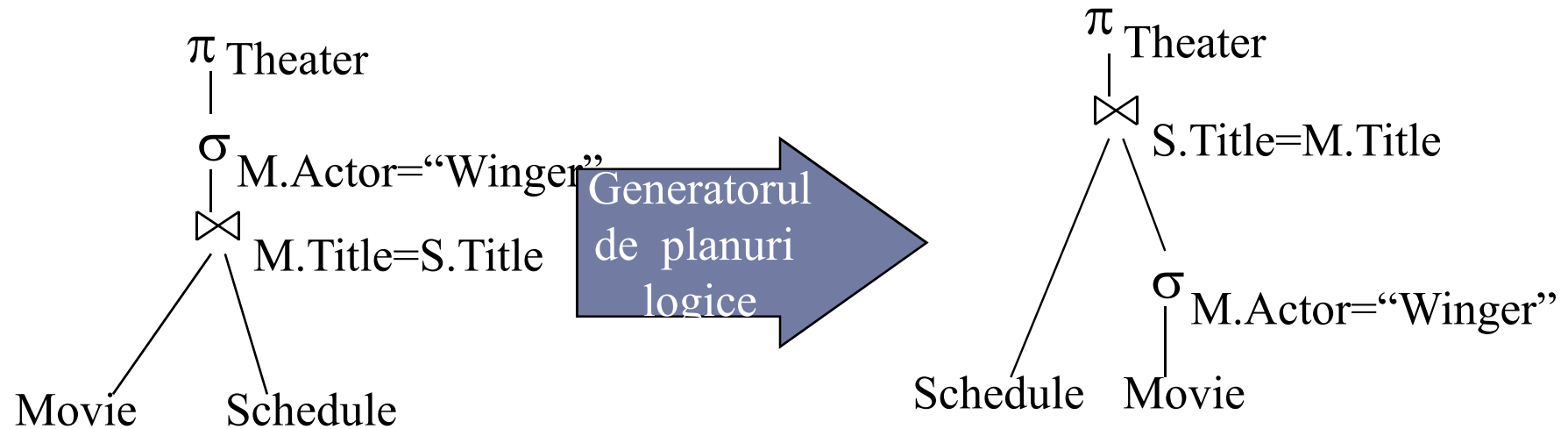
*alt plan logic*



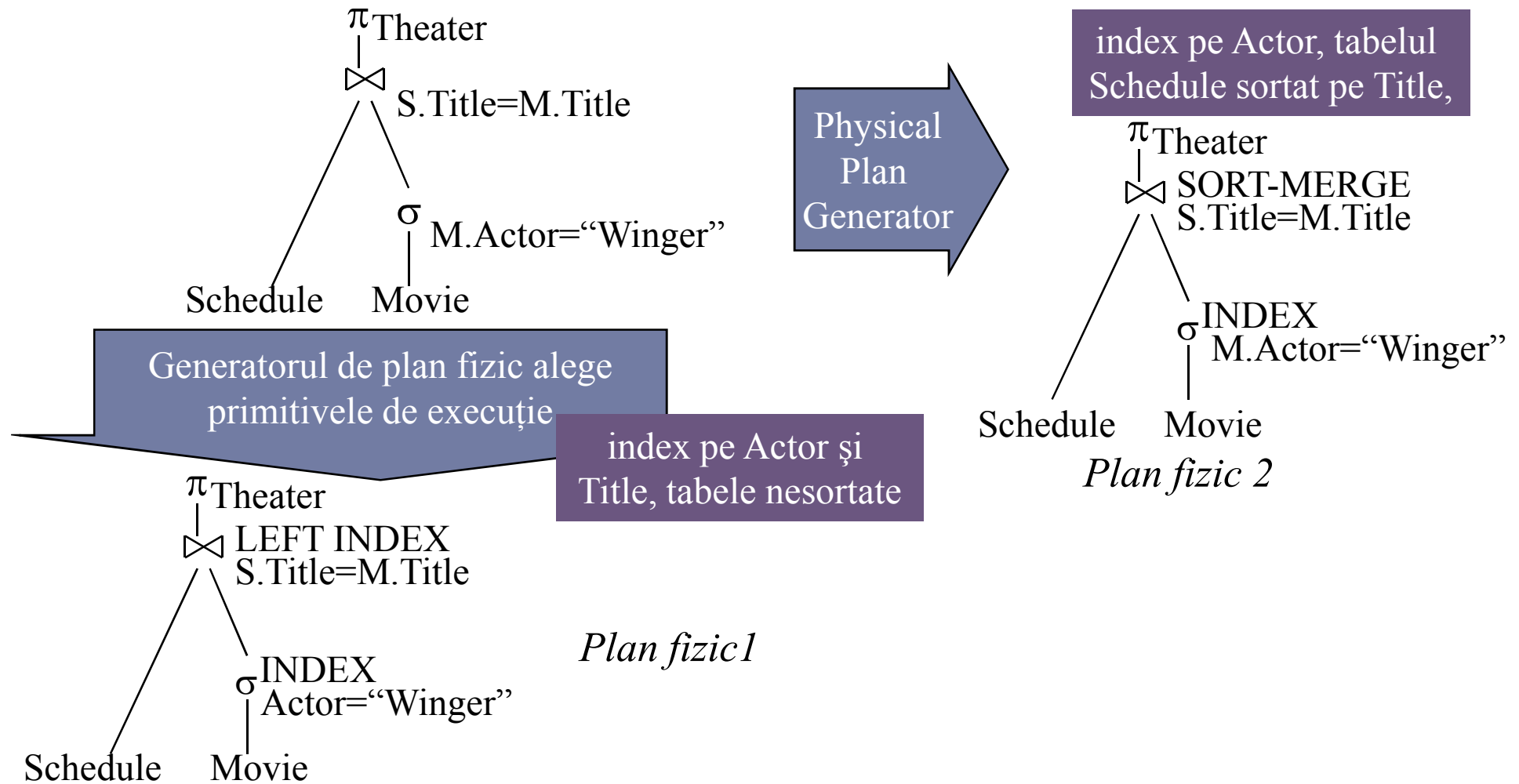
*alt plan logic*



# Drumul unei interogări



# Drumul unei interogări





# Analiza sintactică

- ▶ Gramatică independentă de context

`<query> ::= <SFV> | (<query>)`

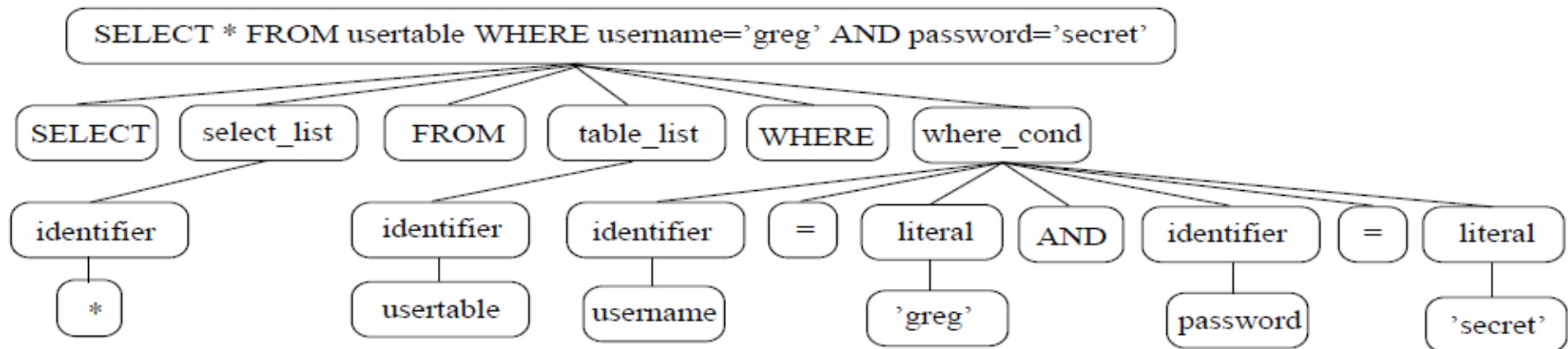
`<SFV> ::= SELECT <select_list> FROM <table_list> WHERE <where_cond>`

`<select_list> ::= <identifier>, <select_list> | <identifier>`

`<table_list> ::= <identifier>, <table_list> | <identifier>`

...

- ▶ Rezultatul parsării: arbore de parsare



- ▶ Gramatica SQL in forma BNF: <http://savage.net.au/SQL/index.html>

# Analiza semantică

## Preprocesare

---

- ▶ Rescrierea apelurilor la view-uri
- ▶ Verificarea relațiilor
- ▶ Verificarea atributelor și a ambiguității
- ▶ Verificarea tipurilor

Dacă arborele de parsare este valid el este transformat într-o expresie cu operatori din algebra relațională

# Operatori în algebra relațională (revizitat)

---

- ▶ Șase operatori de bază
  - ▶ Selecția:  $\sigma$
  - ▶ Proiecția:  $\Pi$
  - ▶ Reuniunea:  $\cup$
  - ▶ Diferența:  $-$
  - ▶ Produsul cartezian:  $\times$
  - ▶ Redenumirea:  $\rho$
- ▶ Operatorii iau ca intrare una sau două relații și generează o nouă relație

# Operatorul de selecție

---

## ► Relația $r$

$A$	$B$	$C$	$D$
$\alpha$	$\alpha$	1	7
$\alpha$	$\beta$	5	7
$\beta$	$\beta$	12	3
$\beta$	$\beta$	23	10

## ► $\sigma_{A=B \wedge D > 5}(r)$

$A$	$B$	$C$	$D$
$\alpha$	$\alpha$	1	7
$\beta$	$\beta$	23	10

# Operatorul de proiecție

## ► Relația $r$

$A$	$B$	$C$
$\alpha$	10	1
$\alpha$	20	1
$\beta$	30	1
$\beta$	40	2

## ► $\Pi_{A,C}(r)$

$A$	$C$
$\alpha$	1
$\alpha$	1
$\beta$	1
$\beta$	2

$A$	$C$
$\alpha$	1
$\beta$	1
$\beta$	2

# Operatorul reuniune

## ► Relațiile $r$ și $s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$r$

A	B
$\alpha$	2
$\beta$	3

$s$

## ► $r \cup s$ :

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1
$\beta$	3

# Operatorul diferență

---

## ► Relațiile $r$ și $s$

$A$	$B$
$\alpha$	$1$
$\alpha$	$2$
$\beta$	$1$

$r$

$A$	$B$
$\alpha$	$2$
$\beta$	$3$

$s$

## ► $r-s$

$A$	$B$
$\alpha$	$1$
$\beta$	$1$

# Produsul cartezian

## ► Relațiile $r$ și $s$

$A$	$B$
-----	-----

$\alpha$	1
$\beta$	2

$r$

$C$	$D$	$E$
-----	-----	-----

$\alpha$	10	$a$
$\beta$	10	$a$
$\beta$	20	$b$
$\gamma$	10	$b$

$s$

## ► $r \times s$

$A$	$B$	$C$	$D$	$E$
-----	-----	-----	-----	-----

$\alpha$	1	$\alpha$	10	$a$
$\alpha$	1	$\beta$	10	$a$
$\alpha$	1	$\beta$	20	$b$
$\alpha$	1	$\gamma$	10	$b$
$\beta$	2	$\alpha$	10	$a$
$\beta$	2	$\beta$	10	$a$
$\beta$	2	$\beta$	20	$b$
$\beta$	2	$\gamma$	10	$b$



# Operatorul de redenumire

---

- ▶  $\rho_X(E)$  - returnează expresia E sub numele X
- ▶ Dacă o expresie E în algebra relațională are aritate n atunci

$$\rho_{X(A_1, A_2, \dots, A_n)}(E)$$

returnează rezultatul expresiei E sub numele X și attributele redenumite în  $A_1, A_2, \dots, A_n$ .

# Compunerea operatorilor

►  $\sigma_{A=C}(r \times s)$

1.  $r \times s$

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	10	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

2.  $\sigma_{A=C}(r \times s)$

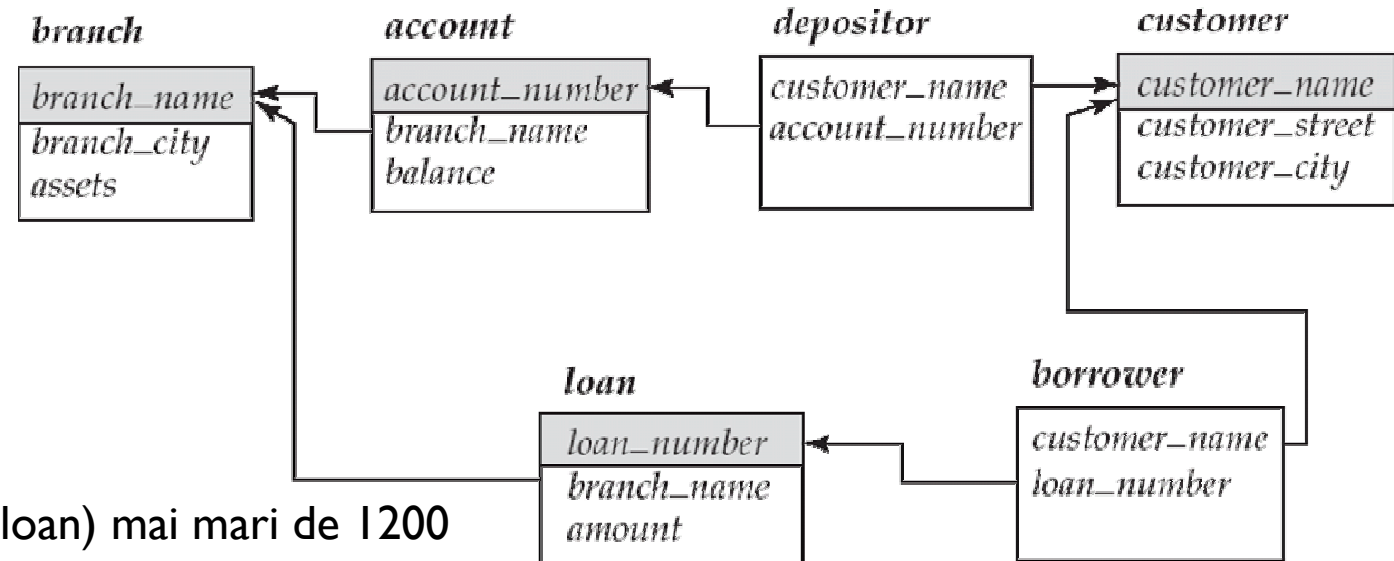
A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b

# Expresii în algebra relațională

---

- ▶ Cea mai simplă expresie este o relație în baza de date
- ▶ Fie  $E_1$  și  $E_2$  expresii în algebra relațională; următoarele sunt expresii în algebra relațională:
  - ▶  $E_1 \cup E_2$
  - ▶  $E_1 - E_2$
  - ▶  $E_1 \times E_2$
  - ▶  $\sigma_P(E_1)$ ,  $P$  este un predicat peste atribute din  $E_1$
  - ▶  $\Pi_S(E_1)$ ,  $S$  este o listă de atribute din  $E_1$
  - ▶  $\rho_x(E_1)$ ,  $x$  este noul nume pentru rezultatul lui  $E_1$

# Exprimarea interogărilor în algebra relațională



- ▶ Împumuturile (loan) mai mari de 1200

$$\sigma_{amount > 1200} (loan)$$

- ▶ Numărul împrumutului (loan\_number) pentru împrumuturi mai mari de 1200

$$\Pi_{loan\_number} (\sigma_{amount > 1200} (loan))$$

- ▶ Numele clienților care au un împrumut, un cont sau ambele la bancă

$$\Pi_{customer\_name} (borrower) \cup \Pi_{customer\_name} (depositor)$$

# Interogări

---

- ▶ Numele tuturor clienților care au un împrumut la filiala Perryridge

$$\Pi_{customer\_name} (\sigma_{branch\_name="Perryridge"} \\ (\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

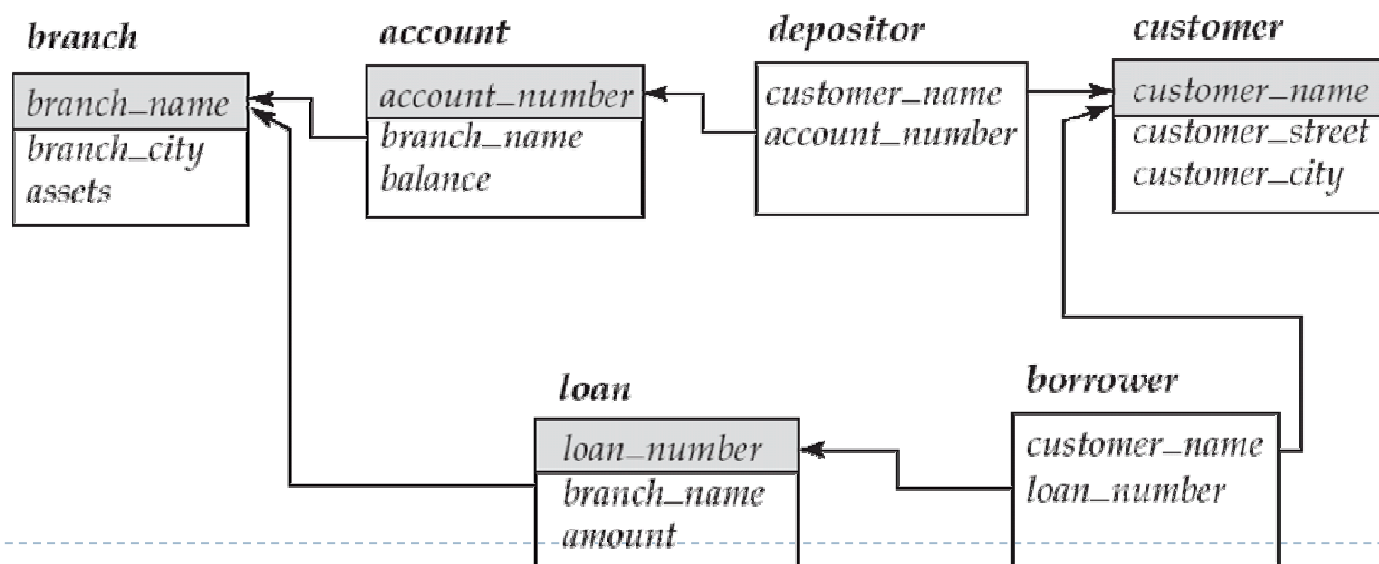
- ▶ Numele tuturor clienților care au un împrumut la filiala Perryridge dar nu au un cont la nici o filială a băncii

$$\Pi_{customer\_name} (\sigma_{branch\_name = "Perryridge"} \\ (\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan))) - \\ \Pi_{customer\_name}(depositor)$$

# Interogări

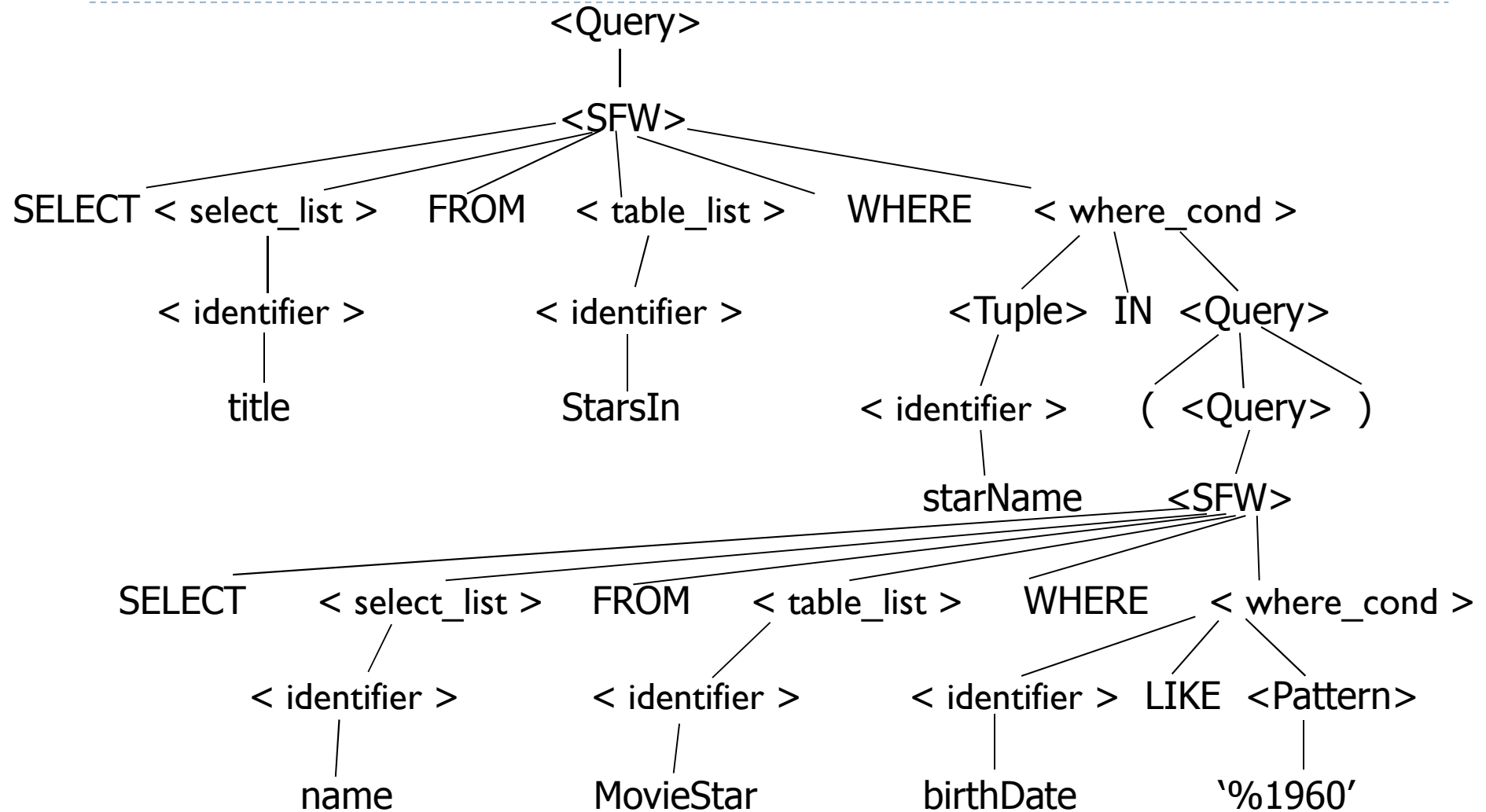
► Numele tuturor clienților care au un împrumut la filiala Perryridge

- $\Pi_{\text{customer\_name}} (\sigma_{\text{branch\_name} = \text{"Perryridge"}} (\sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}} (\text{borrower} \times \text{loan})))$
- $\Pi_{\text{customer\_name}} (\sigma_{\text{loan.loan\_number} = \text{borrower.loan\_number}} (\sigma_{\text{branch\_name} = \text{"Perryridge"}} (\text{loan})) \times \text{borrower}))$



# Conversia unei interogări la compilare

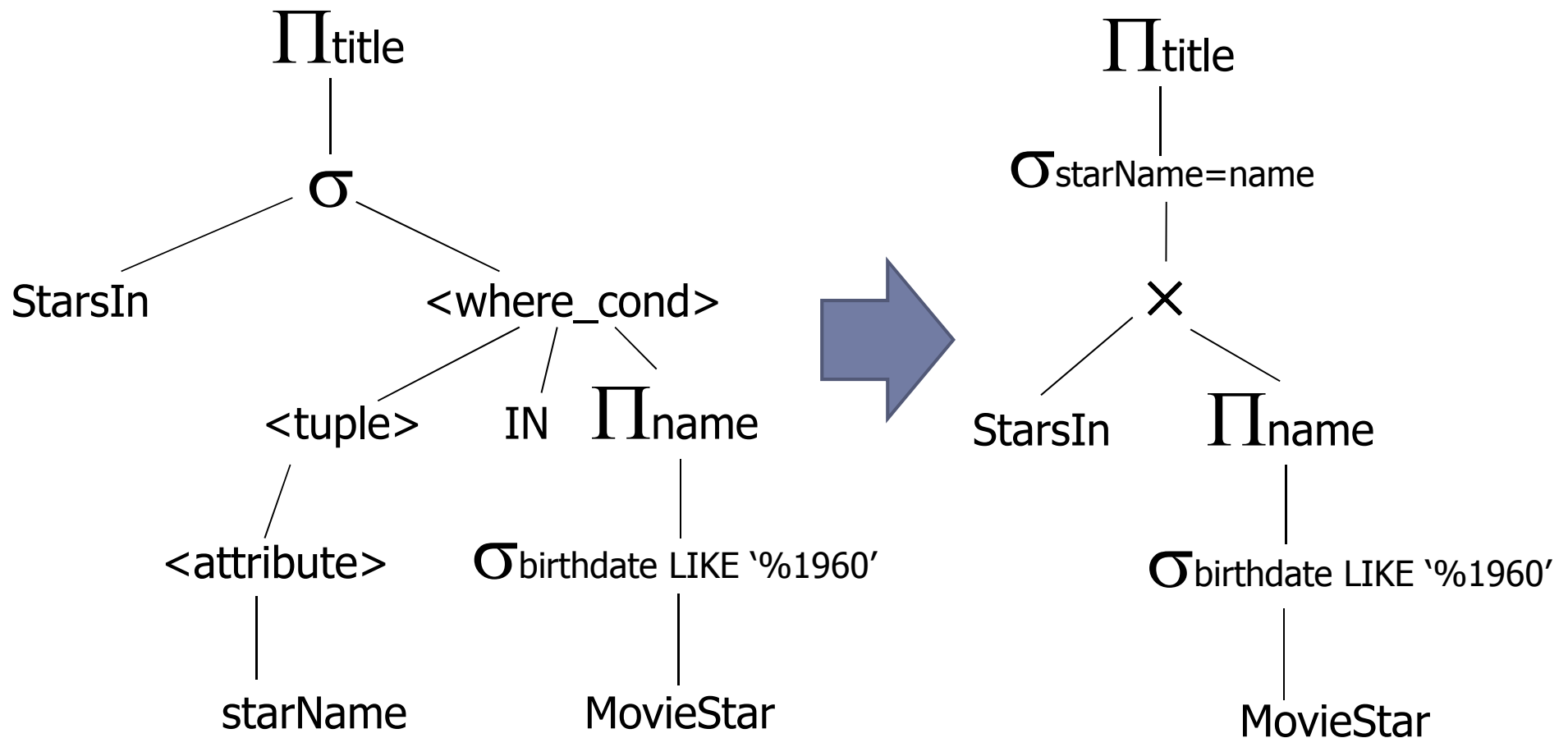
## Arbore de parsare



# Conversia unei interogări la compilare

## Arbore în algebra relațională

---





# Operatori adiționali

---

- ▶ Intersecția pe mulțimi
  - ▶ Joinul natural
  - ▶ Agregarea
  - ▶ Joinul extern
  - ▶ Teta-joinul
- 
- ▶ Toți cu excepția agregării pot fi exprimați utilizând operatori de bază

# Intersecția pe mulțimi

---

## ► Relațiile $r$ și $s$

A	B
$\alpha$	1
$\alpha$	2
$\beta$	1

$r$

A	B
$\alpha$	2
$\beta$	3

$s$

## ► $r \cap s$

A	B
$\alpha$	2

# Joinul natural

► Relațiile r și s

A	B	C	D
$\alpha$	1	$\alpha$	a
$\beta$	2	$\gamma$	a
$\gamma$	4	$\beta$	b
$\alpha$	1	$\gamma$	a
$\delta$	2	$\beta$	b

r

B	D	E
1	a	$\alpha$
3	a	$\beta$
1	a	$\gamma$
2	b	$\delta$
3	b	$\epsilon$

s

►  $r \bowtie s$

A	B	C	D	E
$\alpha$	1	$\alpha$	a	$\alpha$
$\alpha$	1	$\alpha$	a	$\gamma$
$\alpha$	1	$\gamma$	a	$\alpha$
$\alpha$	1	$\gamma$	a	$\gamma$
$\delta$	2	$\beta$	b	$\delta$

►  $\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$

# Exemplu

---

- ▶ Cea mai mare balanță din tabela account

*account*

<i>account_number</i>
<i>branch_name</i>
<i>balance</i>

$$\Pi_{balance}(account) - \Pi_{account.balance}$$

$$(\sigma_{account.balance < d.balance} (account \times \rho_d(account)))$$

# Funcții de agregare și operatori

---

- ▶ Funcții de agregare:

- ▶ **avg**
- ▶ **min**
- ▶ **max**
- ▶ **sum**
- ▶ **count**

- ▶ Operatorul de agregare în algebra relațională

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

- ▶  $E$  – expresie în algebra relațională
- ▶  $G_1, G_2, \dots, G_n$  o listă de attribute de grupare (poate fi goală)
- ▶ Fiecare  $F_i$  este o funcție de agregare
- ▶ Fiecare  $A_i$  este un atribut

# Exemplu

- ▶ relația  $r$

A	B	C
$\alpha$	$\alpha$	7
$\alpha$	$\beta$	7
$\beta$	$\beta$	3
$\beta$	$\beta$	10

- ▶  $g_{\text{sum}(c)}(r)$

sum(c )
27

- ▶ Care operații de agregare nu pot fi exprimate pe baza celorlalți operatori relaționali?

# Join extern

relația loan

<i>loan_number</i>	<i>branch_name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

relația borrower

<i>customer_name</i>	<i>loan_number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

► *loan* ⋈ *borrower* (join natural)

<i>loan_number</i>	<i>branch_name</i>	<i>amount</i>	<i>customer_name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

► *loan* ⋈<sub>l</sub> *borrower* (join extern stânga)

<i>loan_number</i>	<i>branch_name</i>	<i>amount</i>	<i>customer_name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	null

# Join extern

## ➤ Join extern dreapta

*loan* ⋈<sub>□</sub> *borrower*

<i>loan_number</i>	<i>branch_name</i>	<i>amount</i>	<i>customer_name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	<i>null</i>	<i>null</i>	Hayes

## ➤ Join extern plin

*loan* ⋈<sub>□</sub> *borrower*

<i>loan_number</i>	<i>branch_name</i>	<i>amount</i>	<i>customer_name</i>
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	<i>null</i>
L-155	<i>null</i>	<i>null</i>	Hayes



# Exemple interogări

---

- ▶ Numele clienților care au un împrumut și un cont la bancă

$$\Pi_{customer\_name} (borrower) \cap \Pi_{customer\_name} (depositor)$$

- ▶ Numele clienților care au un împrumut la bancă și suma împrumutată

$$\Pi_{customer\_name, loan\_number, amount} (borrower \bowtie loan)$$

- ▶ Clienții care au conturi de la măcar cele două filiale Downtown și Uptown

$$\Pi_{customer\_name} (\sigma_{branch\_name = \text{“Downtown”}} (depositor \bowtie account)) \cap$$

$$\Pi_{customer\_name} (\sigma_{branch\_name = \text{“Uptown”}} (depositor \bowtie account))$$

# Echivalența expresiilor

---

- ▶ Două expresii în algebra relațională sunt echivalente dacă acestea generează același set de upe pe orice instanță a bazei de date
  - ▶ ordinea upelor e irelevantă
- ▶ Obs: SQL lucrează cu multiseturi
  - ▶ în versiunea multiset a algebrei relaționale echivalența se verifică relativ la multiseturi de upe

# Reguli de echivalență

---

1. selecția pe bază de conjuncții e echivalentă cu o secvență de selecții

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. operațiile de selecție sunt comutative

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. într-un șir de proiecții consecutive doar ultima efectuată e necesară

$$\Pi_{L_1}(\Pi_{L_2}(\dots(\Pi_{L_n}(E))\dots)) = \Pi_{L_1}(E)$$

4. selecțiile pot fi combinate cu produsul cartezian și teta joinurile

- a.  $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$

- b.  $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

# Reguli de echivalență

---

5. operațiile de tetra-join și de join natural sunt comutative

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

6. a) Operațiile de join natural sunt asociative

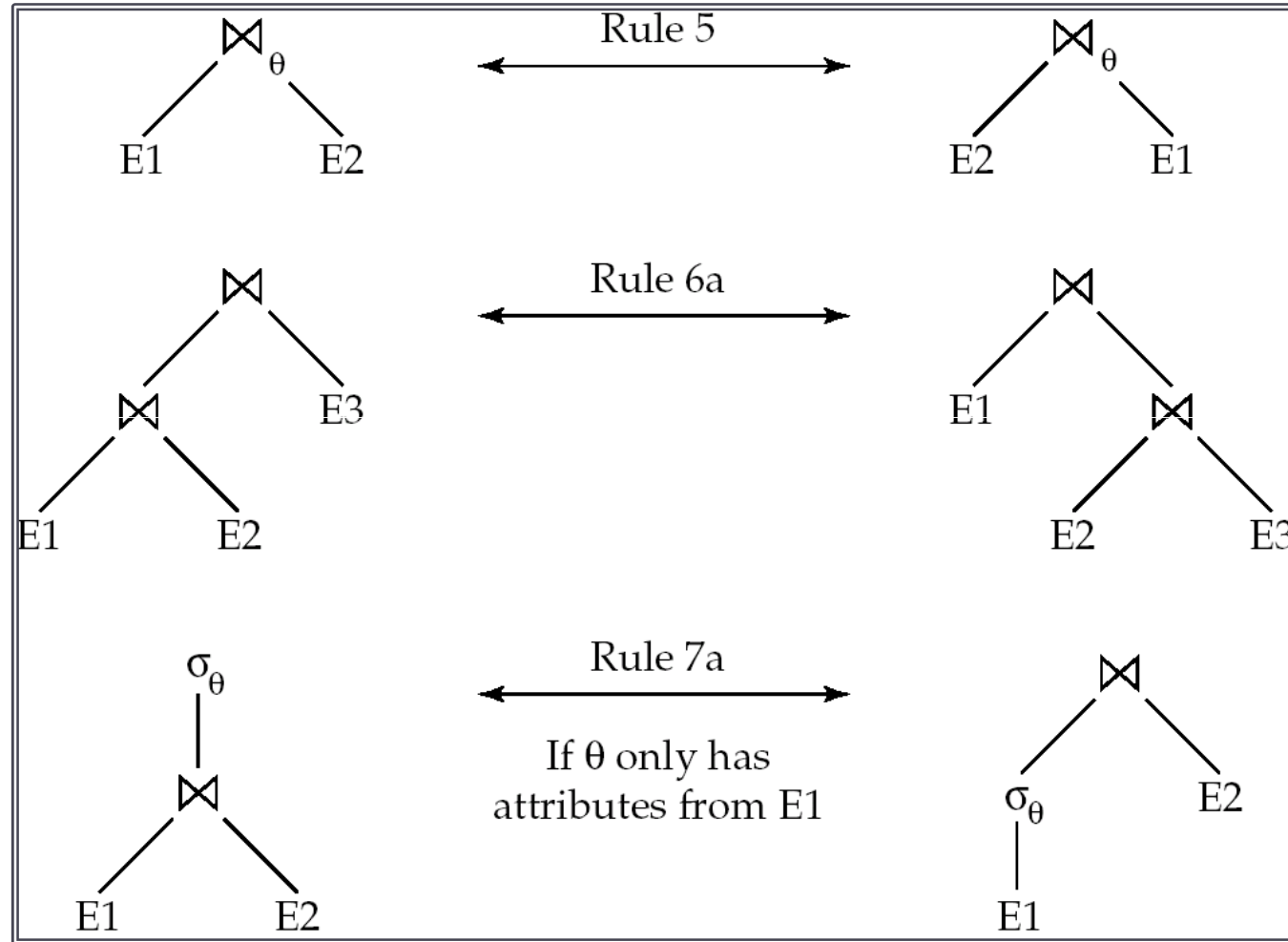
$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

- b) Operațiile de tetra-join sunt asociative astfel:

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

unde  $\theta_2$  implică attribute doar din  $E_2$  și  $E_3$

# Reguli de echivalență



# Reguli de echivalență

---

## 7. Distribuția selecției asupra operatorului de theta-join

- a) când  $\theta_0$  implică attribute doar din una dintre expresiile ( $E_1$ ) din join.

$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$

- b) când  $\theta_1$  implică numai attribute din  $E_1$  și  $\theta_2$  implică numai attribute din  $E_2$ .

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$

# Reguli de echivalență

---

## 8. Distribuția proiecției asupra teta-joinului

a) dacă  $\theta$  implică numai attribute din  $L_1 \cup L_2$ :

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$$

b) Fie joinul  $E_1 \bowtie_{\theta} E_2$

Fie  $L_1$  și  $L_2$  mulțimi de attribute din  $E_1$  și respectiv  $E_2$

Fie  $L_3$  attribute din  $E_1$  care sunt implicate în condiția de join  $\theta$ , dar nu sunt în  $L_1 \cup L_2$ ,

Fie  $L_4$  attribute din  $E_2$  care sunt implicate în condiția de join  $\theta$ , dar nu sunt în  $L_1 \cup L_2$

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\Pi_{L_2 \cup L_4}(E_2)))$$

# Reguli de echivalență

---

9. Operațiile de reuniune și intersecție pe mulțimi sunt comutative

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_1$$

10. Reuniunea și intersecția pe mulțimi sunt asociative

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. Selecția se distribuie peste  $\cup$ ,  $\cap$  și  $-$ .

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2)$$

similar pentru  $\cup$  și  $\cap$  în locul  $-$

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

similar pentru  $\cap$  în locul  $-$ , dar nu pentru  $\cup$

12. Proiecția se distribuie peste reuniune

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$



# Optimizări

## Împingerea selecțiilor

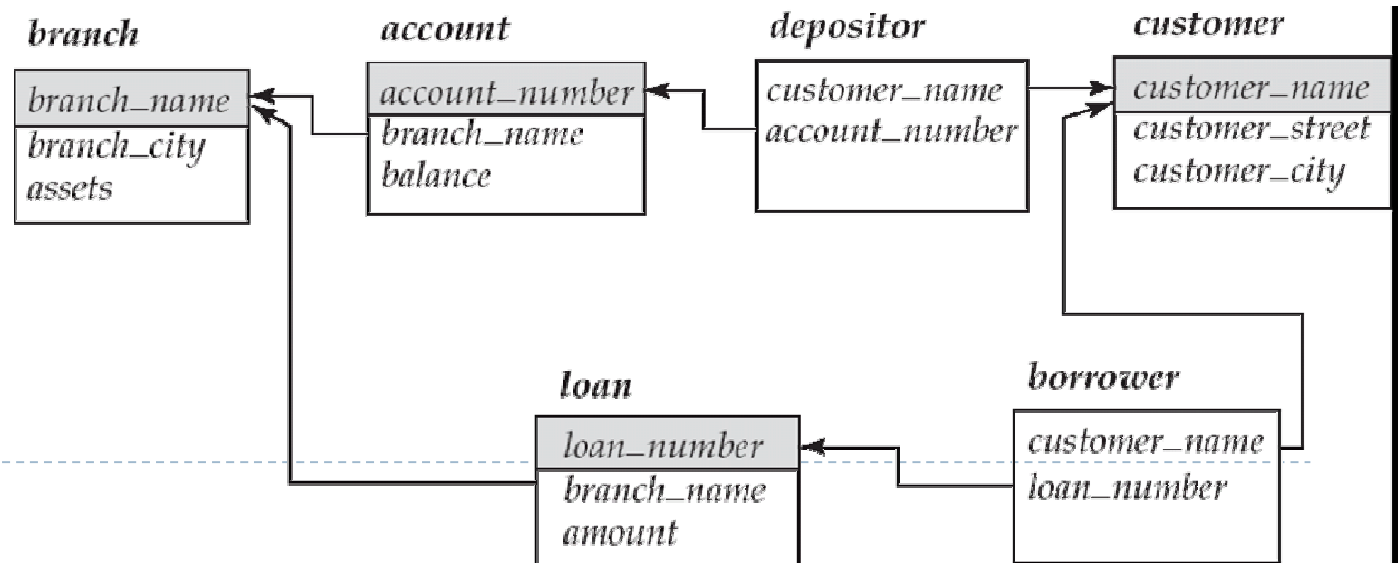
- Numele clienților care au un cont la o filială din Brooklyn

$\Pi_{customer\_name}(\sigma_{branch\_city = \text{"Brooklyn"}}(branch \bowtie (account \bowtie depositor)))$

- Pe baza regulii 7a

$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"}}(branch)) \bowtie (account \bowtie depositor))$

- Realizarea selecției în primele etape reduce dimensiunea relației care participă în join



# Optimizări

## Împingerea selecțiilor

---

- ▶ Numele clienților cu un cont la o filială din Brooklyn care are balanța peste 1000

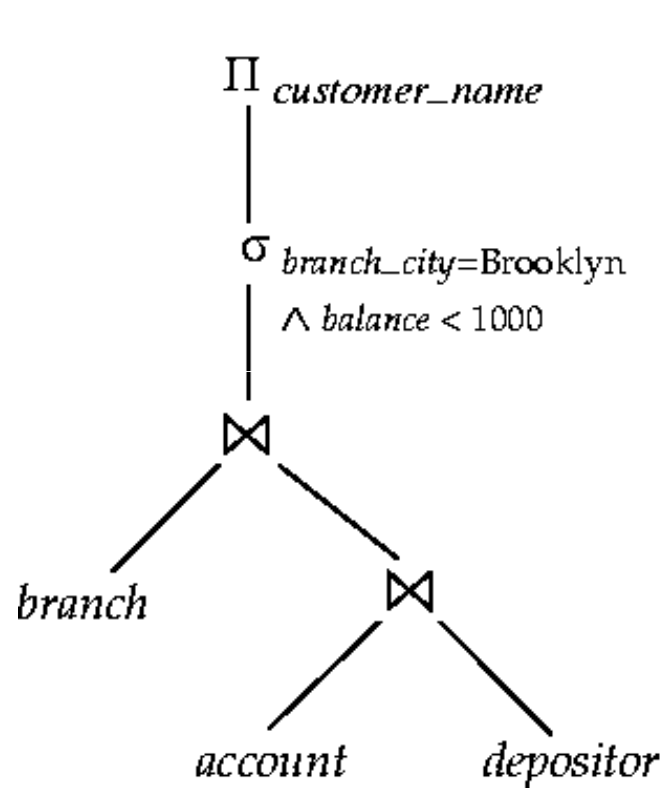
$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{"Brooklyn"} \wedge balance > 1000} (branch \bowtie (account \bowtie depositor)))$$

- ▶ Regula 6a (asociativitatea la join)

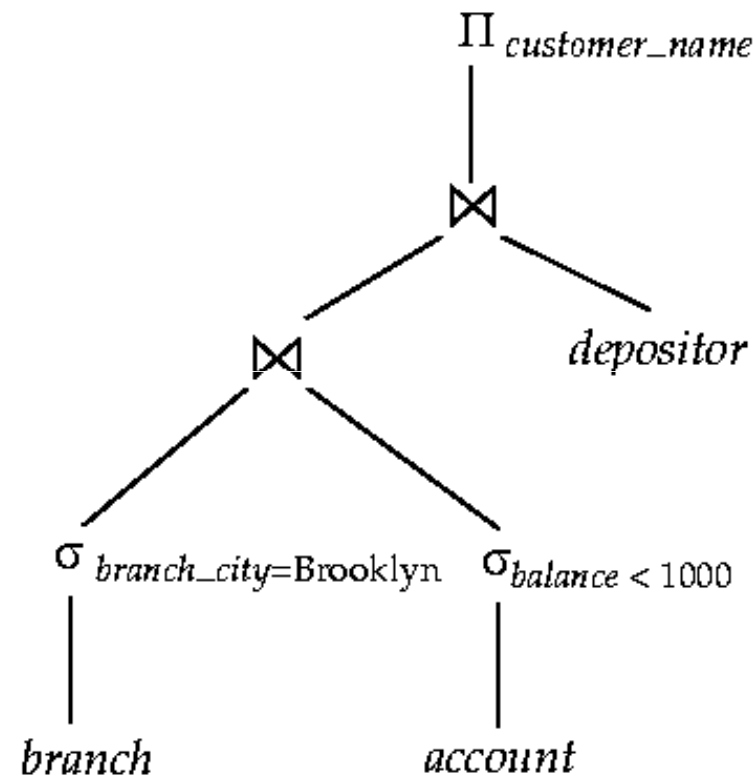
$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{"Brooklyn"} \wedge balance > 1000} (branch \bowtie account)) \bowtie depositor)$$

- ▶ A doua formă furnizează oportunitatea de a efectua selecția devreme

$$\sigma_{branch\_city = \text{"Brooklyn"}} (branch) \bowtie \sigma_{balance > 1000} (account)$$



(a) Initial expression tree



(b) Tree after multiple transformations

# Optimizări

## Împingerea proiecțiilor

---

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{“Brooklyn”}} (branch) \bowtie account) \bowtie depositor)$$

- ▶ Eliminarea atributelor care nu sunt necesare din rezultatele intermediare

$$\Pi_{customer\_name} ((\Pi_{account\_number} (\sigma_{branch\_city = \text{“Brooklyn”}} (branch) \bowtie account) \bowtie depositor))$$

- ▶ Realizarea devreme a proiecției reduce dimensiunea relațiilor din join

# Optimizări

## Ordonarea joinurilor

---

- ▶ Pentru orice relații  $r_1, r_2$ , and  $r_3$ ,

$$(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$$

- ▶ Dacă  $r_2 \bowtie r_3$  are dimensiuni mari și  $r_1 \bowtie r_2$  e de dimensiuni mai mici, alegem

$$(r_1 \bowtie r_2) \bowtie r_3$$

- ▶ Exemplu

$$\Pi_{customer\_name} ((\sigma_{branch\_city = \text{“Brooklyn”}}(branch)) \bowtie (account \bowtie depositor))$$

Numai un mic procent din clienți au conturi în filiale din Brooklyn deci e mai bine să se execute mai întâi

$$\sigma_{branch\_city = \text{“Brooklyn”}}(branch) \bowtie account$$

- ▶ Pentru  $n$  relații există  $(2(n-1))!/(n-1)!$  ordonări diferite pentru join.

- ▶  $n = 7 \rightarrow 665280$ ,  $n = 10 \rightarrow 176$  billion!

- ▶ Pentru a reduce numărul de ordonări supuse evaluării se utilizează programarea dinamică

# Estimarea costurilor

---

- ▶  $n_r$ : numărul de uple în relația  $r$ .
- ▶  $b_r$ : numărul de blocuri conținând uple din  $r$ .
- ▶  $l_r$ : dimensiunea unui uplu din  $r$ .
- ▶  $f_r$ : *factorul de bloc* al lui  $r$  — nr. de uple din  $r$  ce intră într-un bloc
- ▶  $V(A, r)$ : numărul de valori distincte care apar în  $r$  pentru atributul  $A$ ; e echivalent cu dimensiunea proiecției  $\Pi_A(r)$  (pe seturi).
- ▶ Dacă uplele lui  $r$  sunt stocate împreună într-un fișier, atunci:

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

# Estimarea dimensiunii selecției

---

- ▶  $\sigma_{A=v}(r)$ 
  - ▶  $n_r / V(A,r)$  : numărul de înregistrări ce satisfac selecția
  - ▶ pentru atribut cheie: 1
- ▶  $\sigma_{A \leq v}(r)$  (cazul  $\sigma_{A \geq v}(r)$  este simetric)
  - ▶ dacă sunt disponibile  $\min(A,r)$  și  $\max(A,r)$ 
    - ▶ 0 dacă  $v < \min(A,r)$
    - ▶  $n_r \cdot \frac{v - \min(A,r)}{\max(A,r) - \min(A,r)}$  altfel
  - ▶ dacă sunt disponibile histograme se poate rafina estimarea anterioară
  - ▶ în lipsa oricărei informații statistice dimensiunea se consideră a fi  $n_r / 2$ .

# Estimarea dimensiunii selecțiilor complexe

- ▶ Selectivitatea unei condiții  $\theta_i$  este probabilitatea ca un uplu în relația  $r$  să satisfacă  $\theta_i$ 
  - ▶ dacă numărul de uple ce satisfac  $\theta_i$  este  $s_i$ , *selectivitatea* e  $s_i / n_r$
- ▶ Conjuncția (în ipoteza independenței)

$$\sigma_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n}(r): \quad n_r * \frac{s_1 * s_2 * \dots * s_n}{n_r^n}$$

- ▶ Disjuncția

$$\sigma_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n}(r): \quad n_r * \left( 1 - \left( 1 - \frac{s_1}{n_r} \right) * \left( 1 - \frac{s_2}{n_r} \right) * \dots * \left( 1 - \frac{s_n}{n_r} \right) \right)$$

- ▶ Negația

$$\sigma_{\neg \theta}(r): \quad n_r - \text{size}(\sigma_{\theta}(r))$$



# Estimarea dimensiunii joinului

---

- ▶ pentru produsul cartezian  $r \times s$ :  $n_r \cdot n_s$  uple, fiecare uplu ocupă  $s_r + s_s$  octeți
- ▶ pentru  $r \bowtie s$ 
  - ▶  $R \cap S = \emptyset$ :  $n_r \cdot n_s$
  - ▶  $R \cap S$  este o (super)cheie pentru  $R$ :  $\leq n_s$
  - ▶  $R \cap S = \{A\}$  nu e cheie pentru  $R$  sau  $S$ :  $\frac{n_r * n_s}{V(A,s)}$  sau  $\frac{n_r * n_s}{V(A,r)}$ 
    - ▶ minimul este considerat de acuratețe mai mare
    - ▶ dacă sunt disponibile histograme se calculează formulele anterioare pe fiecare celulă pentru cele două relații

# Estimarea dimensiunii pentru alte operații

---

- ▶ Proiecția  $\Pi_A(r) : V(A,r)$
- ▶ Agregarea:  $_A \mathbf{g}_F(r) : V(A,r)$
- ▶ Operații pe mulțimi
  - ▶  $r \cup s : n_r + n_s$
  - ▶  $r \cap s : \min(n_r, n_s)$
  - ▶  $r - s : n_r$
- ▶ Join extern
  - ▶  $r \bowtie s : \dim(r \bowtie s) + n_r$
  - ▶  $r \bowtie \sqsubset s = \dim(r \bowtie s) + n_r + n_s$
- ▶  $\sigma_{\theta_1}(r) \cap \sigma_{\theta_2}(r)$  echivalent cu  $\sigma_{\theta_1} \sigma_{\theta_2}(r)$
- ▶ Estimatorii furnizează în general margini superioare

# Optimizarea planului fizic

# Estimarea costului la nivelul planului fizic

---

- ▶ Costul e în general măsurat ca durata de timp necesară pentru returnarea răspunsului
- ▶ Accesul la disc este costul predominant
  - ▶ Numărul de căutări \*  $t_s$  (timpul pentru o localizare a unui bloc pe disc)
  - ▶ Numărul de blocuri citite/scrise \*  $t_T$  (timpul de transfer)
  - ▶ costul CPU e ignorat pentru simplitate
- ▶ Costul pentru transferul a  $b$  blocuri plus  $S$  căutări:  
$$b * t_T + S * t_s$$

# Algoritmi pentru selectie

---

## ▶ Căutare liniară

- ▶ cost:  $b_r * t_T + t_S$
- ▶ dacă selecția e pe un atribut cheie, costul estimativ:  $b_r/2 * t_T + t_S$
- ▶ poate fi aplicată indiferent de condiția de selecție, ordonarea înregistrărilor în fișier, existența indecșilor

## ▶ Căutarea binară

- ▶ aplicabilă pentru condiții de selecție de tip egalitate pe atributul după care e ordonat fișierul
- ▶ costul găsirii primului uplu ce satisface condiția:  $\lceil \log_2(b_r) \rceil * (t_T + t_S)$ ; dacă există mai multe uple se adaugă timpul de transfer al blocurilor

## ▶ Scanarea indexului – condiția de selecție = cheia de căutare a indexului

- ▶ index primar pe cheie candidat, egalitate:  $(h_i + 1) * (t_T + t_S)$
- ▶ index primar pe non-cheie, egalitate:  $h_i * (t_T + t_S) + t_S + t_T * b$
- ▶ index secundar, egalitate, n uple returnate:  $(h_i + n) * (t_T + t_S)$
- ▶ index primar, comparație:  $h_i * (t_T + t_S) + t_S + t_T * b$

# Algoritmi pentru selecții complexe

---

- ▶ **Conjunție:**  $\sigma_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n}(r)$ 
  - ▶ utilizarea unui index pentru  $\theta_i$  și verificarea celorlalte condiții pe măsură ce uuplele sunt aduse în memorie
  - ▶ utilizarea unui index multi-cheie
  - ▶ intersecția identificatorilor (pointerilor la înregistrări) returnați de indecșii asociați condițiilor urmată de citirea înregistrărilor
- ▶ **Disjuncție:**  $\sigma_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n}(r)$ 
  - ▶ reuniunea identificatorilor

# Algoritmi pentru join

---

- ▶ Algoritmi:
  - ▶ join cu bucle imbricate (nested-loop join)
  - ▶ join indexat cu bucle imbricate
  - ▶ join cu fuziune (merge join)
  - ▶ join hash
- ▶ Alegerea se face pe baza estimării costului
- ▶ Sunt necesare estimări realizate la nivelul planului logic

# Join cu bucle imbricate

---

- ▶ Pentru teta-join:  $r \bowtie_{\theta} s$   
    **for each** uplu  $t_r$  **in**  $r$  **do begin**  
        **for each** uplu  $t_s$  **in**  $s$  **do begin**  
            **if**  $(t_r t_s)$  satisface  $\theta$   
                adaugă  $t_r \cdot t_s$  la rezultat  
        **end**  
    **end**
- ▶ relația interioară –  $s$
- ▶ relația exterioară –  $r$
- ▶ Costul estimat:  $(n_r * b_s + b_r) * t_T + (n_r + b_r) * t_S$



# Join indexat cu bucle imbricate

- ▶ Căutările în index pot înlocui scanarea fișierelor dacă:
  - ▶ e un echi-join sau join natural
  - ▶ există un index pe atributul de join al relației interioare
- ▶ pentru fiecare uplu  $t_r$  în relația exterioară  $r$  se utilizează indexul pentru localizarea uplelor din  $s$  care satisfac condiția de join cu uplul  $t_r$
- ▶ costul:  $b_r (t_T + t_S) + n_r * c$ 
  - ▶  $c$  este costul parcurgerii indexului pentru a returna  $s$  uple care se potrivesc pentru un uplu din  $r$  (echivalent cu selecția pe  $s$  cu condiția de join)
  - ▶ dacă există indecși pentru ambele relații, relația cu mai puține uple va fi preferată drept relație exterioară în join
- ▶ Exemplu
  - ▶  $depositor \bowtie customer$ ,  $depositor$  relație exterioară
  - ▶  $customer$  are asociat un index primar de tip B<sup>+</sup>-arbore pe atributul de join  $customer-name$ , cu 20 intrări pe nod
  - ▶  $customer$ : 10,000 uple,  $depositor$ : 5000 uple
    - ▶ costul:  $100 + 5000 * 5 = 25,100$  blocuri transferate și căutări (corespondentul în joinul neindexat: 2,000, 100 blocuri transferate și = 5100 căutări)

# Join cu fuziune

---

## ▶ Algoritm

1. se sortează ambele relații în funcție de atributul de join
2. are loc fuziunea relațiilor

## ▶ Poate fi utilizat doar pentru echi-joinuri și joinuri naturale

## ▶ Costul:

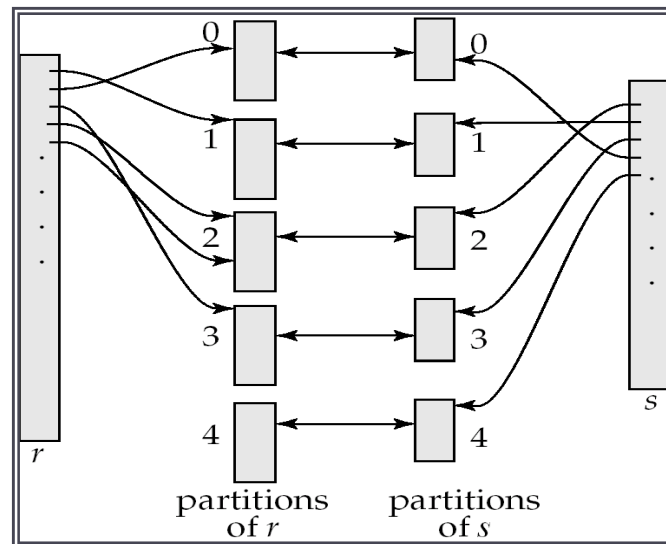
- ▶  $b_r + b_s$  blocuri transferate
- ▶ + costul sortării relațiilor

## ▶ Join cu fuziune hibrid: o relație este sortată iar a doua are un index secundar pe atributul de join de tip B<sup>+</sup>-arbore

- ▶ relația sortată fuzionează cu intrările de pe nivelul frunză al arborelui
- ▶ se sortează rezultatul după adresele uplelor relației nesortate
- ▶ se scanează relația nesortată în ordinea adreselor fizice și se realizează fuziunea cu rezultatul anterior pentru a înlocui adresele cu uplele asociate

# Join hash

- ▶ aplicabil pentru echi-join și join natural
- ▶ o funcție hash  $h$  ce ia la intrare attributele de join partiționează upele ambelor relații în blocuri ce încap în memorie
  - ▶  $r_1, r_2, \dots, r_n$
  - ▶  $s_1, s_2, \dots, s_n$
- ▶ upele din  $r_i$  sunt comparate doar cu upele din  $s_i$



# Joinuri complexe

---

- ▶ Condiție de tip conjuncție:  $r \bowtie_{\theta_1 \wedge \theta_2 \wedge \dots \wedge \theta_n} s$ 
  - ▶ bucle imbricate cu verificarea tuturor condițiilor sau
  - ▶ se calculează un join mai simplu  $r \bowtie_{\theta_i} s$  și se realizează selecția pentru celelalte condiții
- ▶ Condiție de tip disjuncție:  $r \bowtie_{\theta_1 \vee \theta_2 \vee \dots \vee \theta_n} s$ 
  - ▶ bucle imbricate cu verificarea condițiilor sau
  - ▶ calculul reuniunii joinurilor individuale (aplicabil numai versiunii set a reuniunii)  
 $(r \bowtie_{\theta_1} s) \cup (r \bowtie_{\theta_2} s) \cup \dots \cup (r \bowtie_{\theta_n} s)$

# Evaluare expresiilor

---

## ► Alternative:

- Materializarea: (sub)expresiile sunt materializate sub forma unor relații stocate pe disc pentru a fi date ca intrare operatorilor de pe nivele superioare
- Pipelining: uple sunt date ca intrare operațiilor de pe nivele superioare imediat ce acestea sunt returnate în timpul procesării unui operator
  - nu e întotdeauna posibil (sortare, join hash)
  - varianta la cerere: nivelul superior solicită noi uple
  - varianta la producător: operatorul scrie în buffer uple iar părintele scoate din buffer (la umplerea bufferului există timpi de așteptare)

# Bibliografie

---

- ▶ Capitolele 13 și 14 în *Avi Silberschatz Henry F. Korth S. Sudarshan. “Database System Concepts”*. McGraw-Hill Science/Engineering/Math; 4th edition