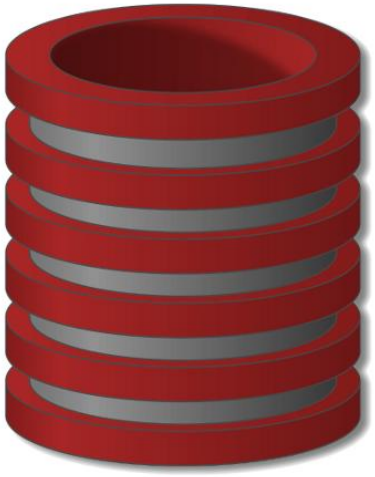


BAZE DE DATE

Procesarea interogărilor

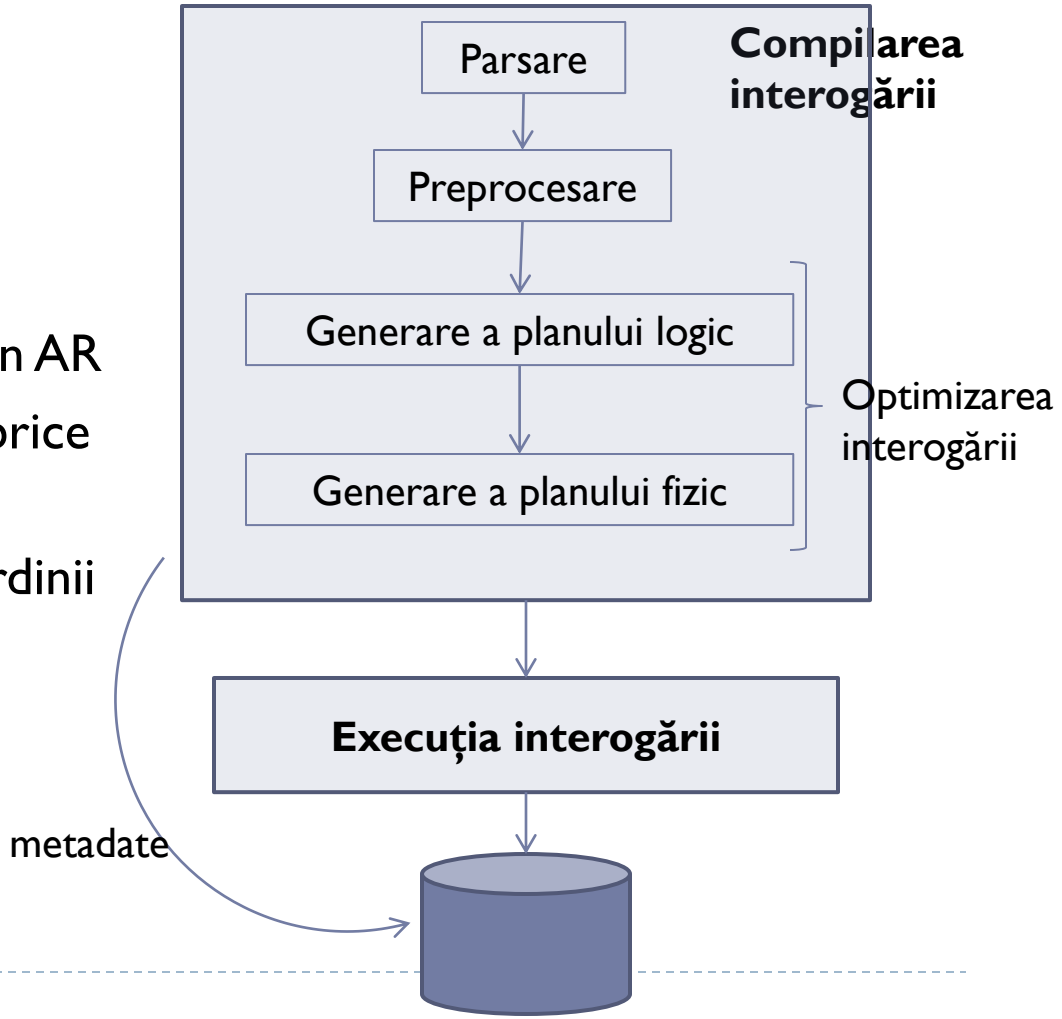
Mihaela Elena Breabăn

Cuprins

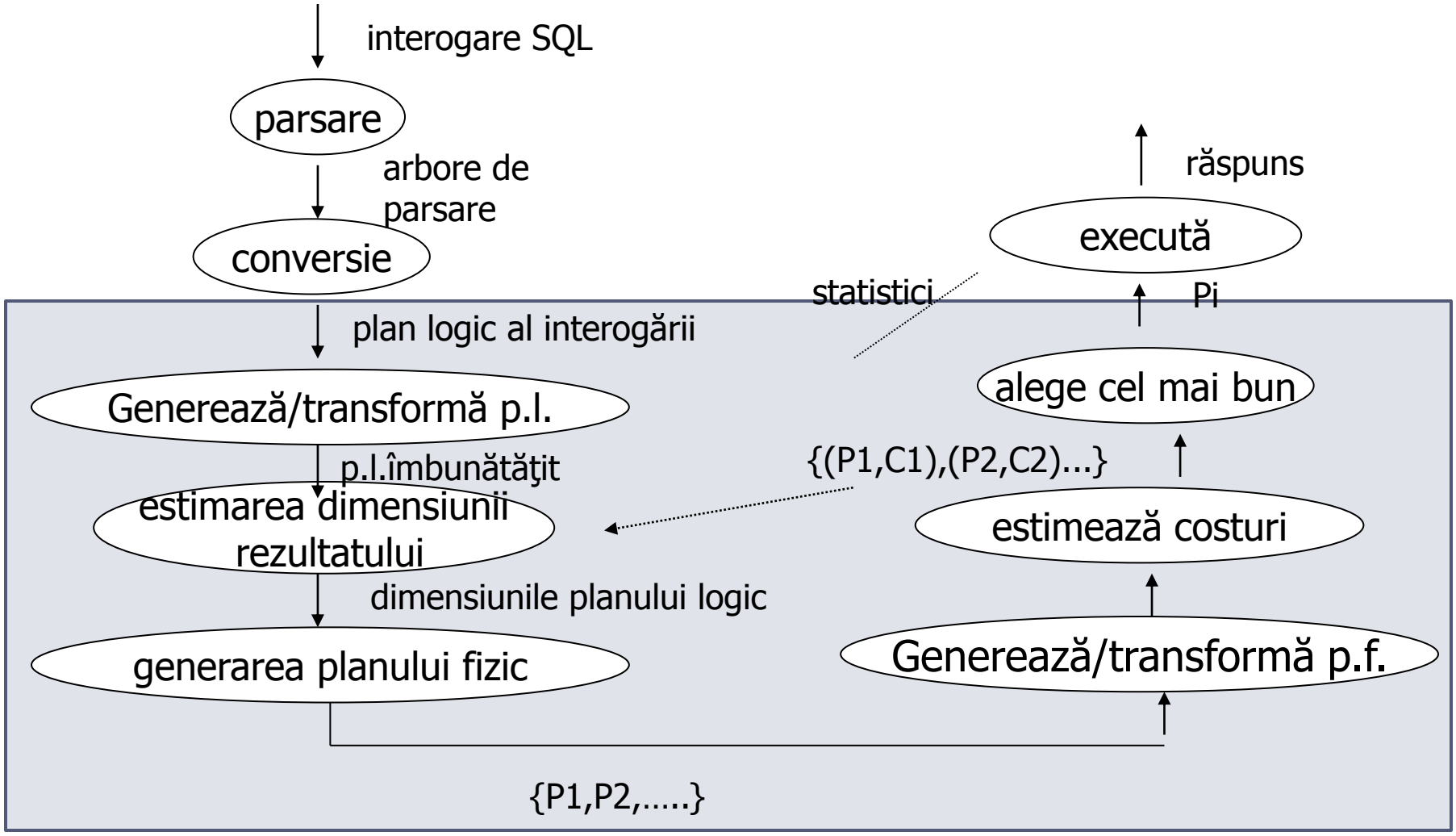
- Etapele procesării interogărilor
- Expresii în algebra relațională
  - Operatori (revizitat)
  - Expresii
  - Echivalența expresiilor
- Estimarea costului interogării
- Algoritmi pentru evaluarea operatorilor/expresiilor în algebra relațională

Etapele procesării interogărilor

- Compilarea interogării
  - Analiza sintactică
    - Parsare
      - Arbore de parsare
  - Analiza semantică
    - Preprocesare si rescriere în AR
    - Selecția reprezentării algebrice
      - Plan logic
    - Selecția algoritmilor și a ordinii
      - Plan fizic



Optimizarea interogărilor



Analiza sintactică

- Gramatică independentă de context

  <query> ::= <SFW> | (<query>)

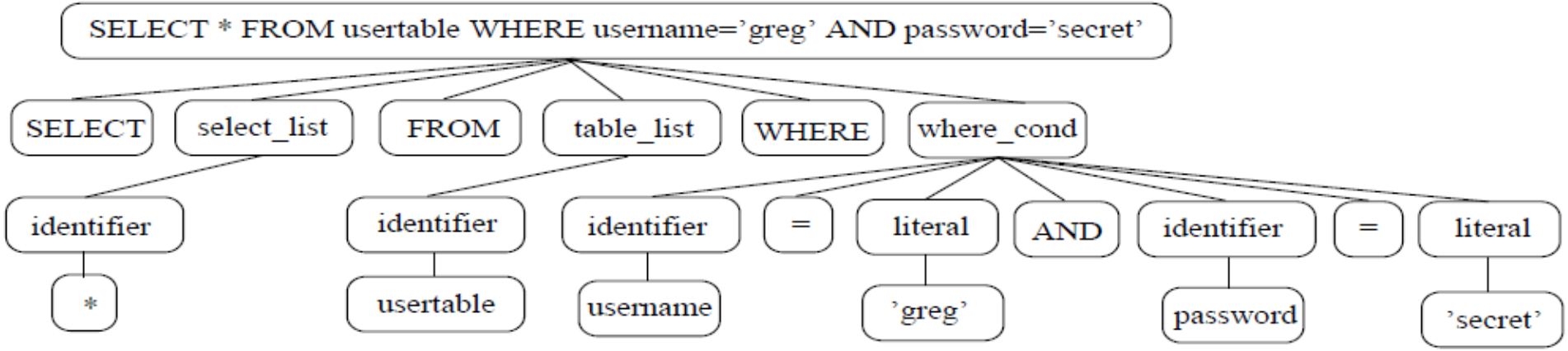
  <SFW> ::= SELECT <select_list> FROM <table_list> WHERE <where_cond>

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

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

  …

- Rezultatul parsării: arbore de parsare



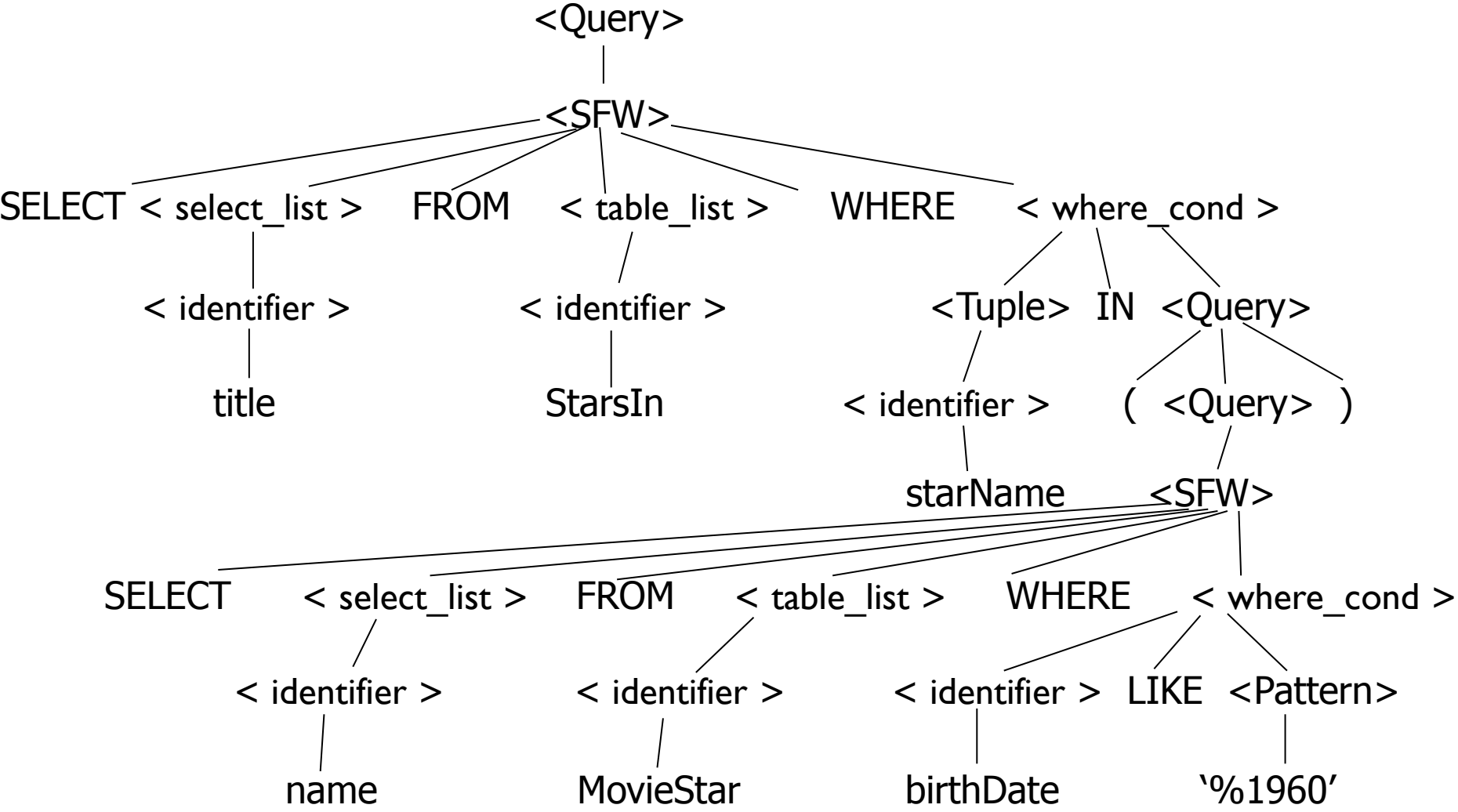
- Gramatica SQL in forma BNF: http://savage.net.au/SQL/index.html

Analiza semantică
Preprocesare

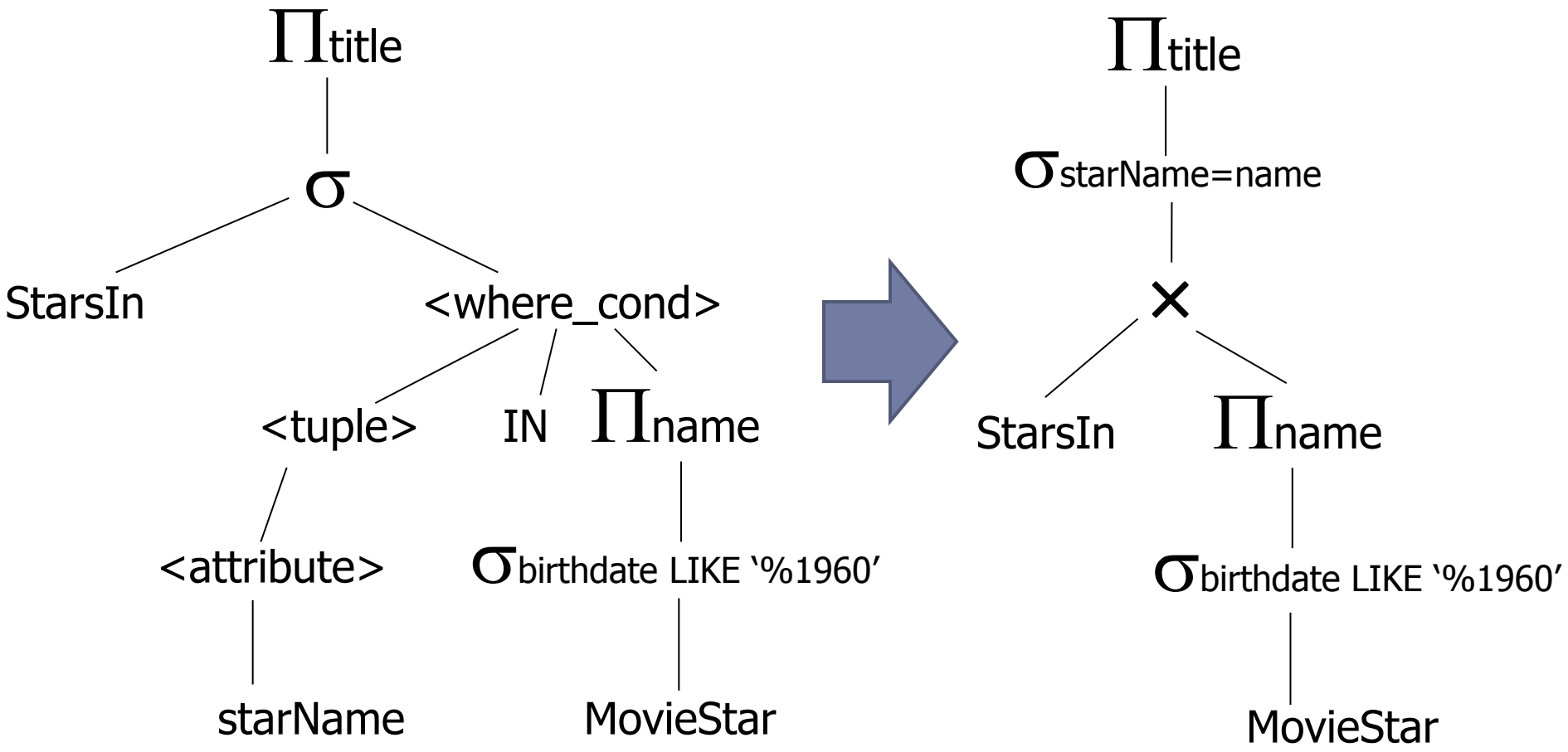
- Rescrierea apelurilor la view-uri
- Verificarea relațiilor
- Verificarea atributelor și a ambiguității
- Verificarea tipurilor

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

Analiza semantică
Rescriere în AR (1)

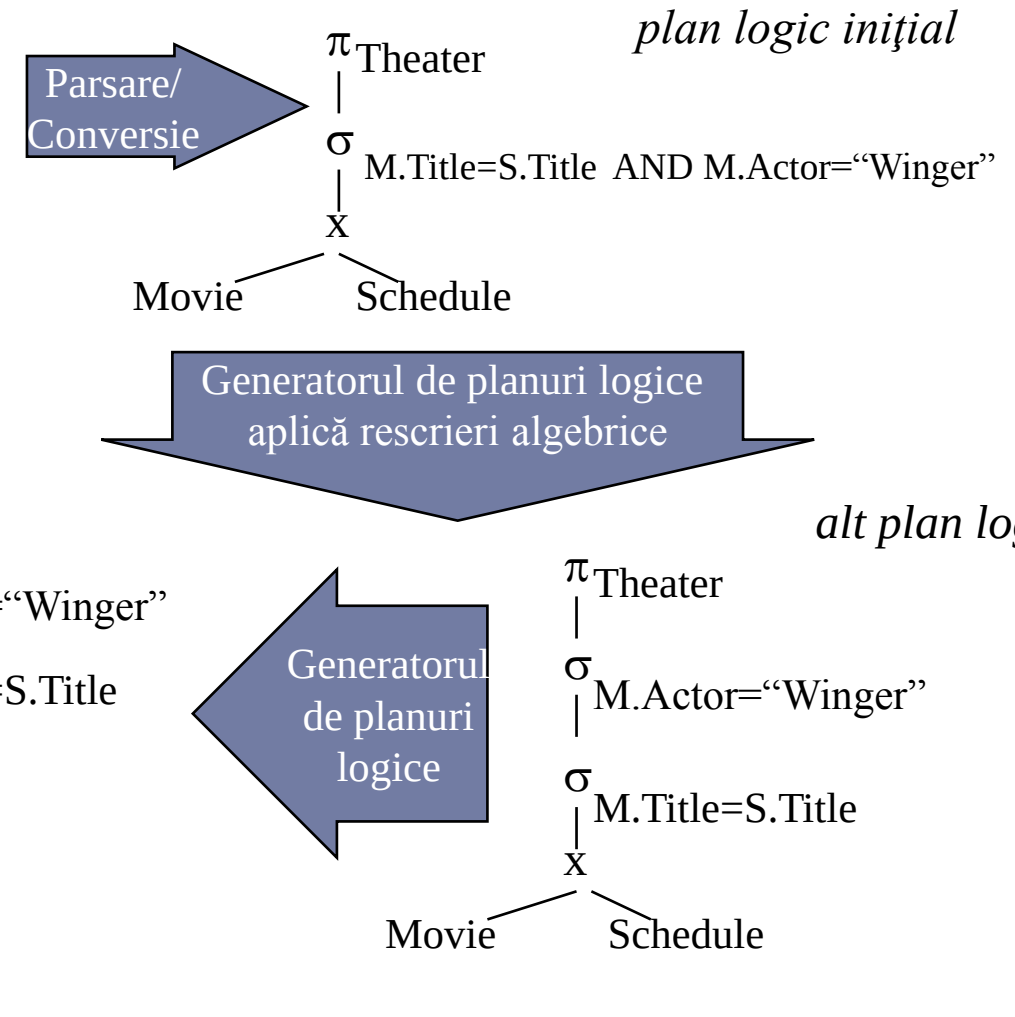


Analiza semantică
Rescriere în AR (2)

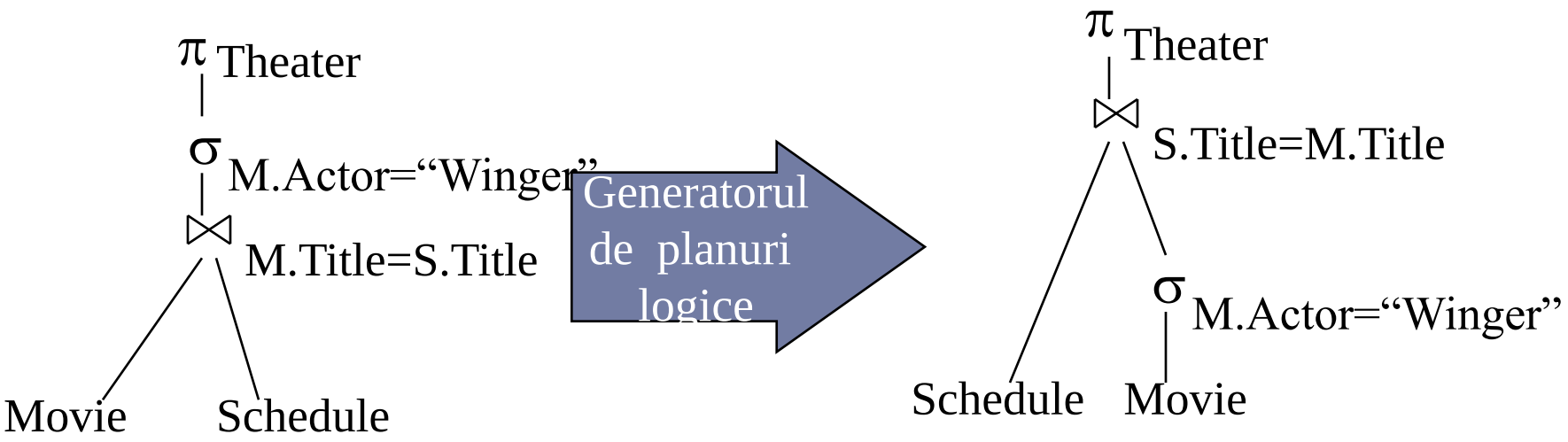


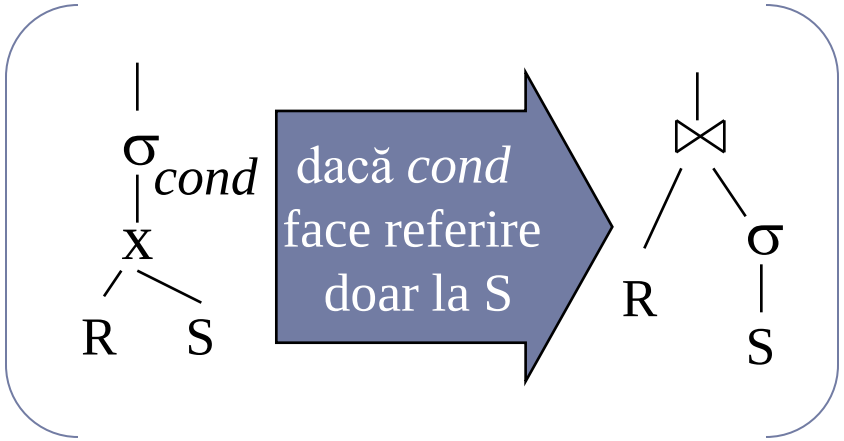
Analiza semantică
Optimizarea planului logic

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

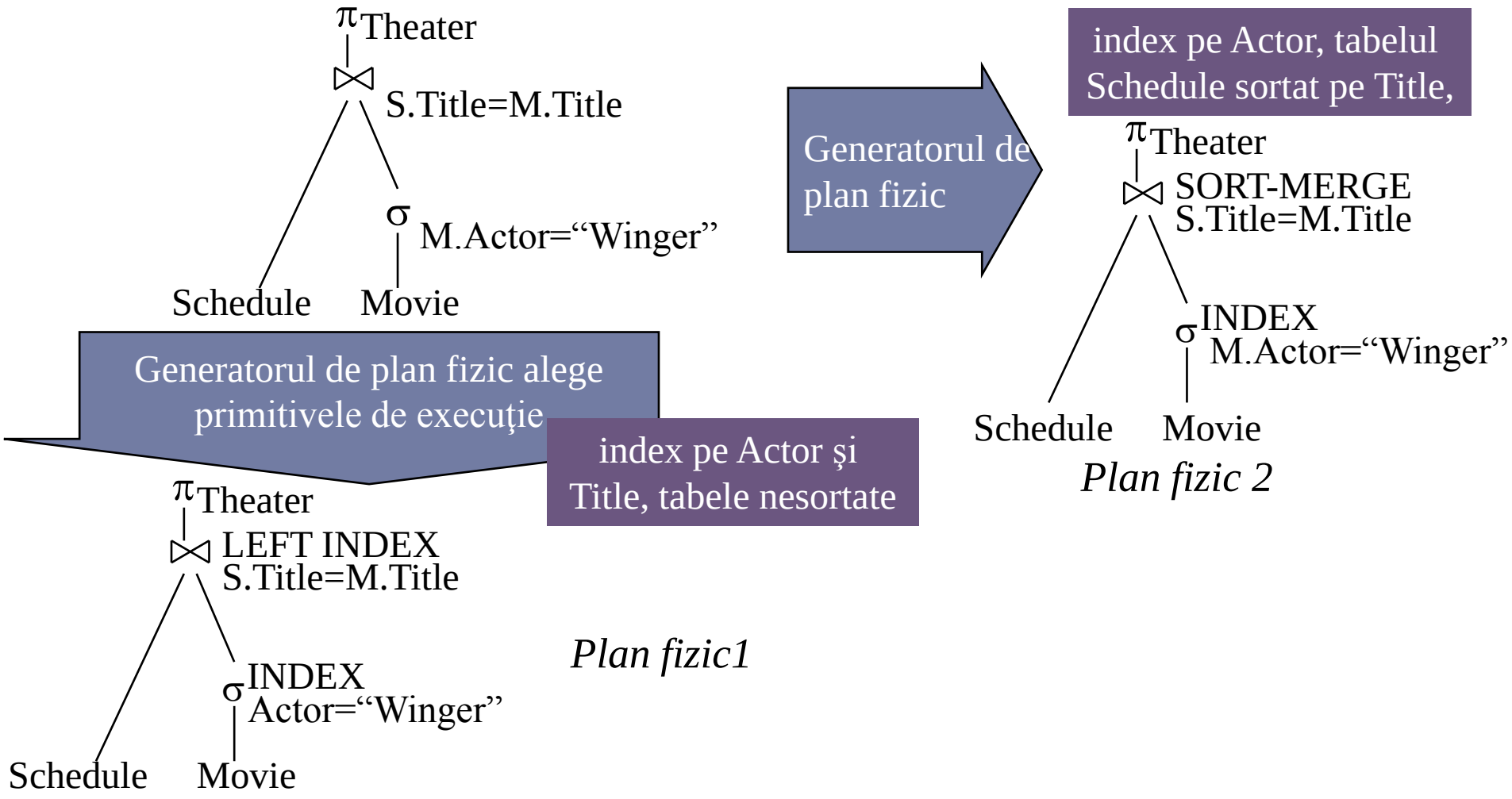


Analiza semantică
Optimizarea planului logic





Analiza semantică
Optimizarea planului fizic



Operatori în algebra relațională (revizitat)

- Șase operatori de bază
  - Selecția: σ
  - Proiecția: $\prod$
  - Reuniunea: $\cup$
  - Diferența: $-$
  - Produsul cartezian: x
  - Redenumirea: ρ
- Operatorii iau ca intrare una sau două relații și generează o noua relație

Operatorul de selecție

- Realația r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Operatorul de proiecție

- Relația r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

A	C
α	1
β	1
β	2

Operatorul reuniune

- Relațiile r și s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r $\cup$ s:

A	B
α	1
α	2
β	1
β	3

Operatorul diferență

- Relațiile r și s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r-s

A	B
α	1
β	1

Produsul cartezian

- Relațiile r și s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	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

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

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

Compunerea operatorilor

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

1. $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

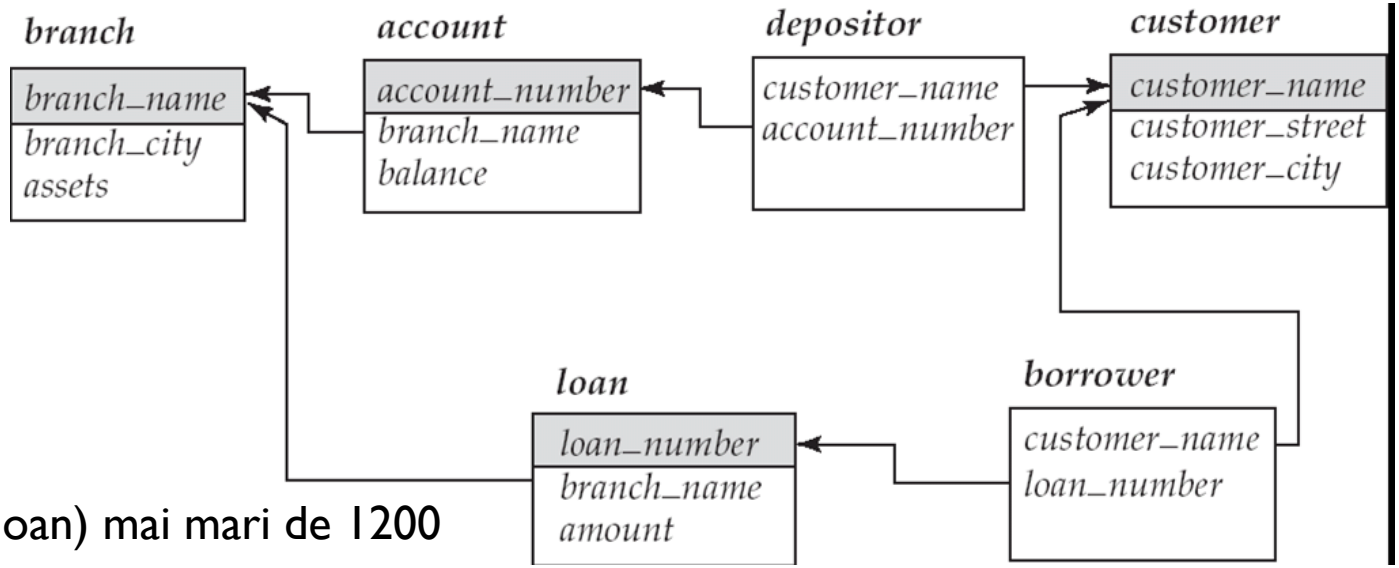
2. $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Expresii în algebra relațională

- Cea mai simpla expresie este o relație în baza de date
- Fie E_1 și E_2 expresii în algebra relațională; următoarele sunt expresii în algebra relatională:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P este un predicat peste atribute din E_1
  - $\prod_s (E_1)$, 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 depozit sau ambele la bancă

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

Interogări

- Numele tuturor clienților care au un împrumut la filiala Perryridge

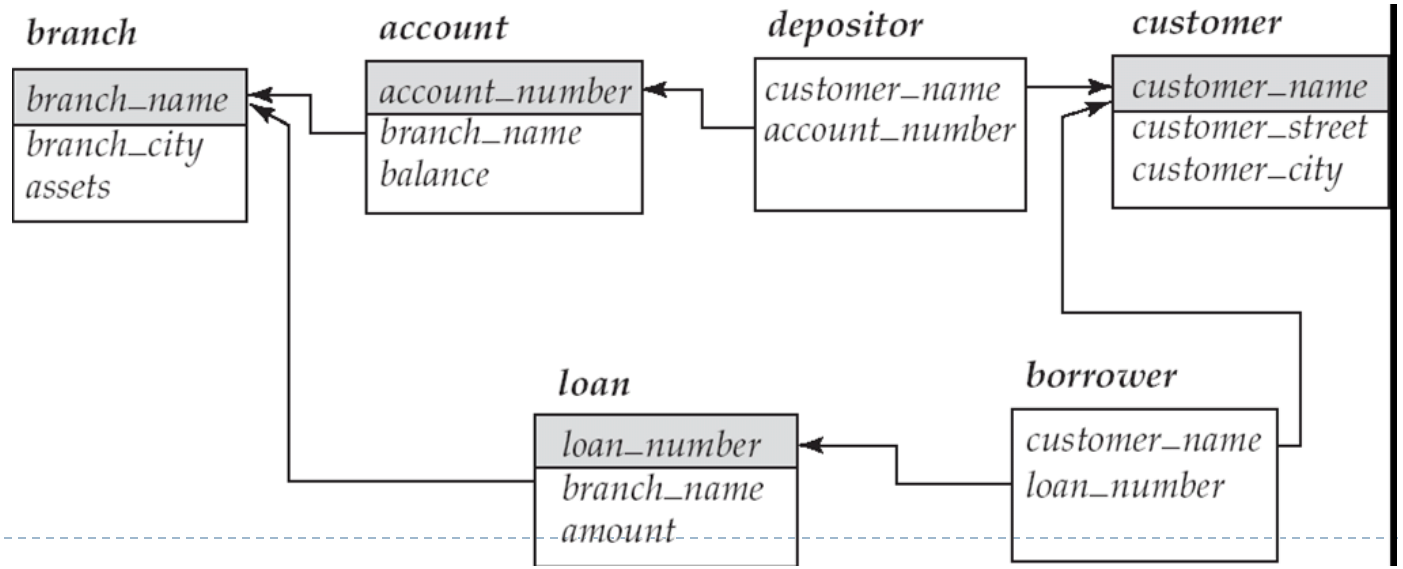
$$\Pi_{customer\_name}\,(\sigma_{branch\_name=\text{“Perryridge”}}\,(\sigma_{borrower.loan\_number\,=\,loan.loan\_number}(borrower\ x\ loan)))$$

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

$$\Pi_{customer\_name}\,(\sigma_{branch\_name\,=\,\text{“Perryridge”}}\,(\sigma_{borrower.loan\_number\,=\,loan.loan\_number}(\text{borrower x loan}))) - \Pi_{customer\_name}(\text{depositor})$$

Interogări

- Numele tuturor clienților care au un împrumut la filiala Perryridge
  - $\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}($
    $\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$
  - $\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}($
    $(\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$



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
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Joinul natural

- Relațiile r și s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	∈

s

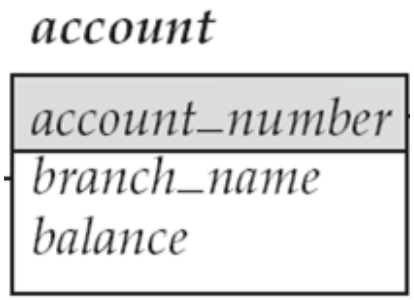
- $r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

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

Agregare
Exemplu

- Cea mai mare balanță din tabela account



$$\Pi_{balance}(account) - \Pi_{account.balance}$$
$$(\sigma_{account.balance < d.balance} (account \times \rho_{d} (account)))$$

Funcții de agregare şi operatori

- Funcții de agregare:
  - **avg**
  - **min**
  - **max**
  - **sum**
  - **count**
  - **var**
- Operatorul de agregare în algebra relațională

$$_{G_1,G_2,\ldots,G_n}\vartheta_{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(E)$$

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

Agregare Exemplu

- relația r

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\mathbf{sum(c)}}(r)$

sum(c)
27

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

Join extern

relația loan

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

relația borrower

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

- *loan* ⋈ *borrower (join natural)*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- *loan* ⟕ *borrower (join extern stânga)*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Join extern

- Join extern dreapta

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

- Join extern plin

  loan ⟗ *borrower*

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

Exemple interogări

- Numele clienților care au un împrumut și un depozit 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 depozite 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 tuple pe orice instanță a bazei de date
  - ordinea tuplelor e irelevantă

- Obs: SQL lucrează cu multiseturi
  - în versiunea multiset a algebrei relaționale echivalența se verifică relativ la multiseturi de tuple

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

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

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 teta-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

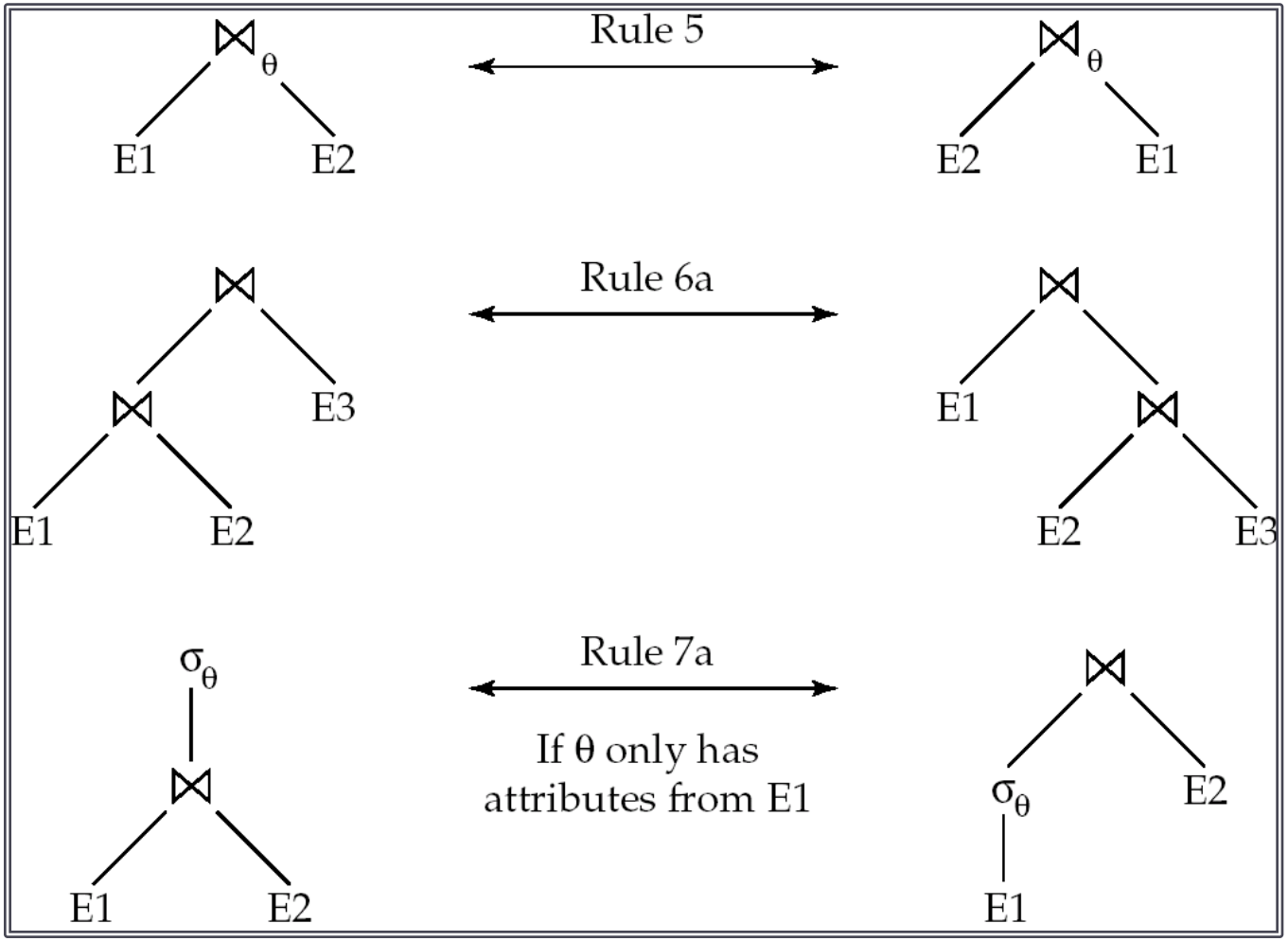
$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

b) Operațiile de teta-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 θ_2 implică atribute doar din E_2 și E_3

Reguli de echivalență



Reguli de echivalență

7. Distribuția selecției asupra operatorului de teta-join

a) când θ_0 implică atribute 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 θ_1 implică numai atribute din E_1 și θ_2 implică numai atribute 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 θ implică numai atribute 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 atribute din E_1 și respectiv E_2

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

Fie L_4 atribute dinE_2 care sunt implicate în condiția de join θ, 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 $\cap$ in 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

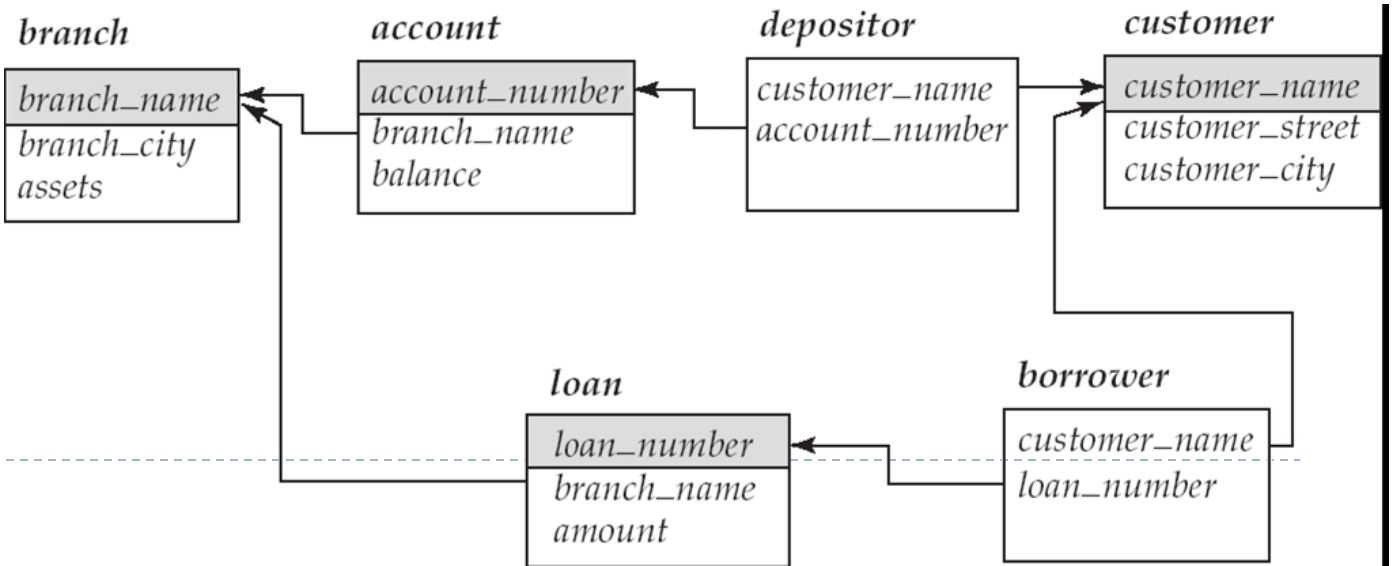
- Numele clienților care au un cont la o filială din Brooklyn

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

- 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

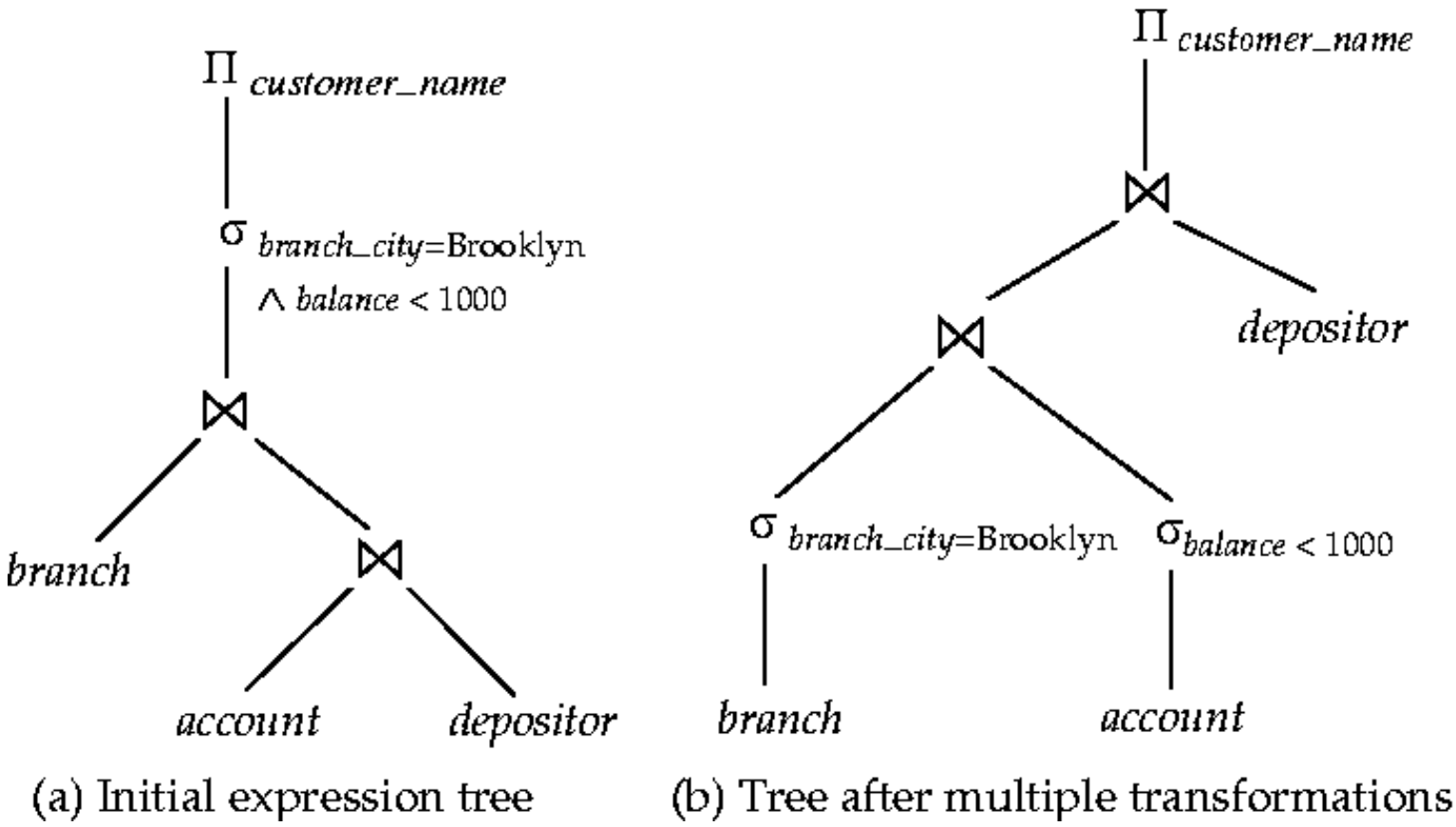
$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{“Brooklyn”} \wedge balance > 1000}(branch \bowtie (account \bowtie depositor)))$$

- 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

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


Π customer_name
σ branch_city=Brooklyn
∧ balance < 1000
branch
account
depositor
(a) Initial expression tree
Π customer_name
depositor
σ branch_city=Brooklyn
σ balance < 1000
branch
account
(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, si 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.
  - $n = 7$ -> 665280, $n = 10$ ->176 billion!
- Pentru a reduce numărul de ordonări supuse evaluării se utilizează programarea dinamică

Estimarea costurilor

- l_r: dimensiunea unui tuplu din *r*.
- n_r: numărul de tuple în relația *r*.
- b_r: numărul de blocuri conținând tuple din *r*.
- f_r: *factorul de* bloc al lui *r* — nr. de tuple din r ce intră într-un bloc
- Dacă tuplele lui *r* sunt stocate împreună într-un fișier, atunci:

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

- *V(A, r)*: numărul de valori distincte care apar in *r* pentru atributul *A*; e echivalent cu dimensiunea proiecției $\prod_A(r)$ (pe seturi).

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 θ_i este probabilitatea ca un tuplu în relația r să satisfacă θ_i
  - dacă numărul de tuple ce satisfac θ_i este s_i , *selectivitatea* e s_i / n_r.
- Conjuncția (în ipoteza independenței)

$\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$: $n_r * \frac{s_1 * s_2 * \ldots * s_n}{n_r^n}$

- Disjuncția

$\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(r)$: $n_r * \left(1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) * \ldots * (1 - \frac{s_n}{n_r})\right)$

- Negația

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

Estimarea dimensiunii joinului

- pentru produsul cartezian r x s: $n_r * n_s$ tuple, fiecare tuplu ocupă $s_r + s_s$ octeți
- pentru $r \bowtie s$
  - $R \cap S = \emptyset$: $n_r * n_s$
  - $R \cap S$ este o (super)cheie pentru R: <= 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: ${}_A\boldsymbol{g}_F(r)$: $V(A,r)$
- Operații pe mulțimi
  - $r \cup s$: $n_r + n_s$.
  - $r \cap s$: $\min(n_r, n_s)$
  - r-s : n_r
- Join extern
  - $r \supset\!\bowtie s$: $dim(r \bowtie s) + n_r$
  - $r \supset\!\bowtie\subset s = dim(r \bowtie s) + n_r + n_s$
- $\sigma_{\theta 1}(r) \cap \sigma_{\theta 2}(r)$ 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 (full scan)
  - 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 tuplu ce satisface condiția: $\lceil \log_2(b_r) \rceil * (t_T + t_S)$; dacă există mai multe tuple 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 tuple 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

- Conjuncție: $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \theta n}(r)$
  - utilizarea unui index pentru θ_i și verificarea celorlalte condiții pe măsură ce tuplele sunt aduse în memorie
  - utilizarea unui index multi-cheie
  - intersecția identificatorilor (pointerilor la înregistrări) returnați de indecșii asociați condițiilor urmată de citirea înregistrărilor
- Disjuncție: $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \theta n}(r)$
  - 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 tuplu t_r **in** r **do begin**
  for each tuplu t_s **in** s **do begin**
  if (t_r, t_s) satisface θ
  adaugă $t_r \bullet t_s$ 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

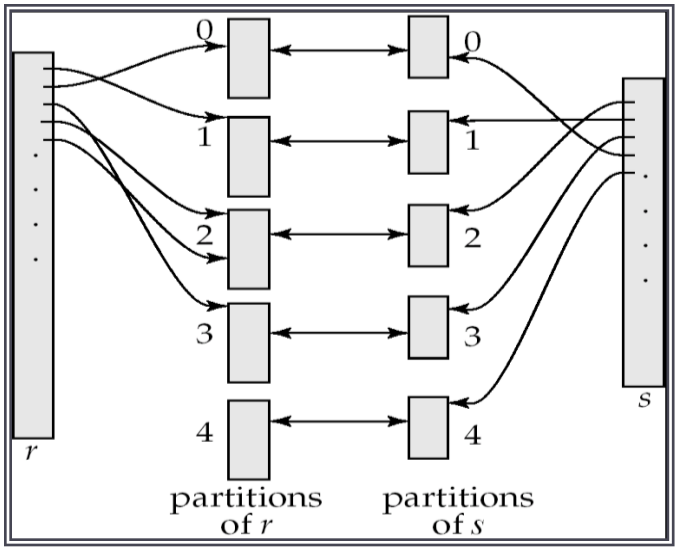
- Căutările în index pot înlocui scanarea fișierelor dacă:
  - e un echi-join sau join natural
  - există un index pe atributul de join al relației interioare
- pentru fiecare tuplu t_r în relația exterioară r se utilizează indexul pentru localizarea tuplelor din s care satisfac condiția de join cu uplul t_r.
- costul: $b_r\ (t_T + t_S) + n_r * c$
  - c este costul parcurgerii indexului pentru a returna tuple din s care se potrivesc pentru un tuplu din r (echivalent cu selecția pe s cu condiția de join)
  - dacă există indecși pentru ambele relații, relația cu mai puține tuple va fi preferată drept relație exterioară în join
- Exemplu
  - *depositor* $\bowtie$ *customer*, *depositor* relație exterioară
  - customer are asociat un index primar de tip B^+-arbore pe atributul de join *customer-name,* cu 20 intrări pe nod
  - *customer:* 10,000 tuple(f=25), *depositor*:5000 tuple (f=50)
    - 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
- 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^+-arbore
  - relația sortată fuzionează cu intrările de pe nivelul frunză al arborelui
  - se sortează rezultatul după adresele tuplelor relației nesortate
  - se scanează relația nesortată în ordinea adreselor fizice și se realizează fuziunea cu rezultatul anterior pentru a înlocui adresele cu tuplele asociate

Join hash

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



Joinuri complexe

- Condiție de tip conjuncție: $r \bowtie_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n} s$
  - 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: $r \bowtie_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n} s$
  - bucle imbricate cu verificarea condițiilor sau
  - calculul reuniunii joinurilor individuale (aplicabil numai versiunii set a reuniunii)

  $(r \bowtie_{\theta 1} s) \cup (r \bowtie_{\theta 2} s) \cup \ldots \cup (r \bowtie_{\theta n} s)$

Eliminarea duplicatelor

- Sortarea tuplelor sau hashing
- Fiindca e costisitoare, SGBD-urile nu elimina duplicatele decat la cerere

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: tuple 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 tuple
    - varianta la producător: operatorul scrie în buffer tuple iar părintele scoate din buffer (la umplerea bufferului există timpi de așteptare)
- Ex: Numele clientilor care au depozite >2000

Planuri de executie
Oracle

- Inregistreaza planul:

EXPLAIN PLAN
[SET STATEMENT_ID = <*id*>]
[INTO <table_name>]
FOR <sql_statement>;

  - Pentru orice comanda DML

- Vizualizeaza planul:

SELECT * FROM table(dbms_xplan.display);

sau

select * from plan_table [where statement_id = <id>];

http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-explain-the-explain-plan-052011-393674.pdf

Planuri de executie
Oracle-statistici

- Table statistics
  - Number of rows
  - Number of blocks
  - Average row length
- Column statistics
  - Number of distinct values (NDV) in column
  - Number of nulls in column
  - Data distribution (histogram)
- Index statistics
  - Number of leaf blocks
  - Levels
  - Clustering factor
- System statistics
  - I/O performance and utilization
  - CPU performance and utilization

Planuri de executie
Colectarea statisticilor

- Proceduri Oracle din pachetul DBMS_STATS:

- GATHER_INDEX_STATS
  - Index statistics
- GATHER_TABLE_STATS
  - Table, column, and index statistics
- GATHER_SCHEMA_STATS
  - Statistics for all objects in a schema
- GATHER_DATABASE_STATS
  - Statistics for all objects in a database
- GATHER_SYSTEM_STATS
  - CPU and I/O statistics for the system

- http://docs.oracle.com/cd/B10500_01/server.920/a96533/stats.htm

Planuri de executie
Hints

- In cadrul unei comenzi DML este posibil a instrui optimizatorul Oracle asupra planului de executie:

```
SELECT /*+ USE_MERGE(employees departments) */ * FROM employees,
    departments WHERE employees.department_id =
    departments.department_id;
```

http://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements006.htm

Bibliografie

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