NỘI DUNG

4.1 Giới thiệu về xử lý truy vấn

4.2 Xử lý truy vấn trong môi trường tập trung

4.3 Xử lý truy vấn trong môi trường phân tán

4.4 Tối ưu hoá truy vấn trong CSDL phân tán

MỤC ĐÍCH

- Giới thiệu một bức tranh tổng quát của bộ tối ưu hóa truy vấn trong môi trường tập trung và phân tán
- Trình bày các quy trình xử lý truy vấn trong hệ thống phân tán

4.1 GIỚI THIỆU VỀ XỬ LÝ TRUY VẤN

Mục đích của xử lý truy vấn:

- Giảm thiểu thời gian xử lý
- Giảm vùng nhớ trung gian
- Giảm chi phí truyền thông giữa các trạm.

Chức năng của xử lý truy vấn:

- Biến đổi một truy vấn ở mức cao thành một truy vấn tương đương ở mức thấp hơn.
- Phép biến đổi này phải đạt được cả về tính đúng đắn và hiệu quả
- Mỗi cách biến đổi dẫn đến việc sử dụng tài nguyên máy tính khác nhau, nên vấn đề đặt ra là lựa chọn phương án nào dùng tài nguyên ít nhất.

4.1 GIỚI THIỆU VỀ XỬ LÝ TRUY VẤN

Các phương pháp xử lý truy vấn cơ bản

- Phương pháp biến đổi đại số:

  Đơn giản hóa câu truy vấn nhờ các phép biến đổi đại số tương đương nhằm giảm thiểu thời gian thực hiện các phép toán, phương pháp này không quan tâm đến kích thước và cấu trúc dữ liệu

- Phương pháp ước lượng chi phí:

  Xác định kích thước dữ liệu, thời gian thực hiện mỗi phép toán trong câu truy vấn. Phương pháp này phải xác định kích thước dữ liệu và chi phí thời gian thực hiện mỗi phép toán trong câu truy vấn

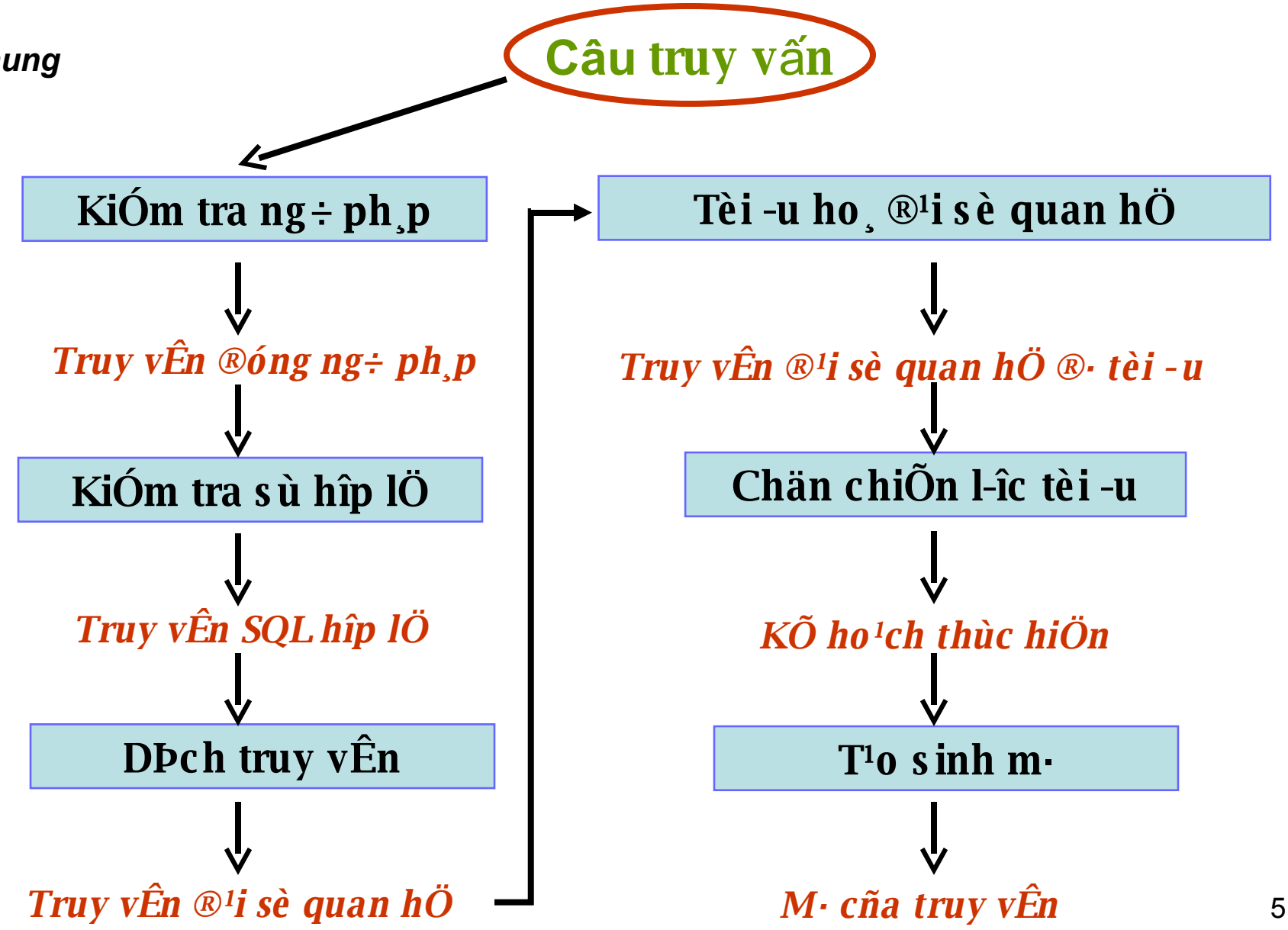
4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

4.2.1 So sánh xử lý truy vấn tập trung và phân tán

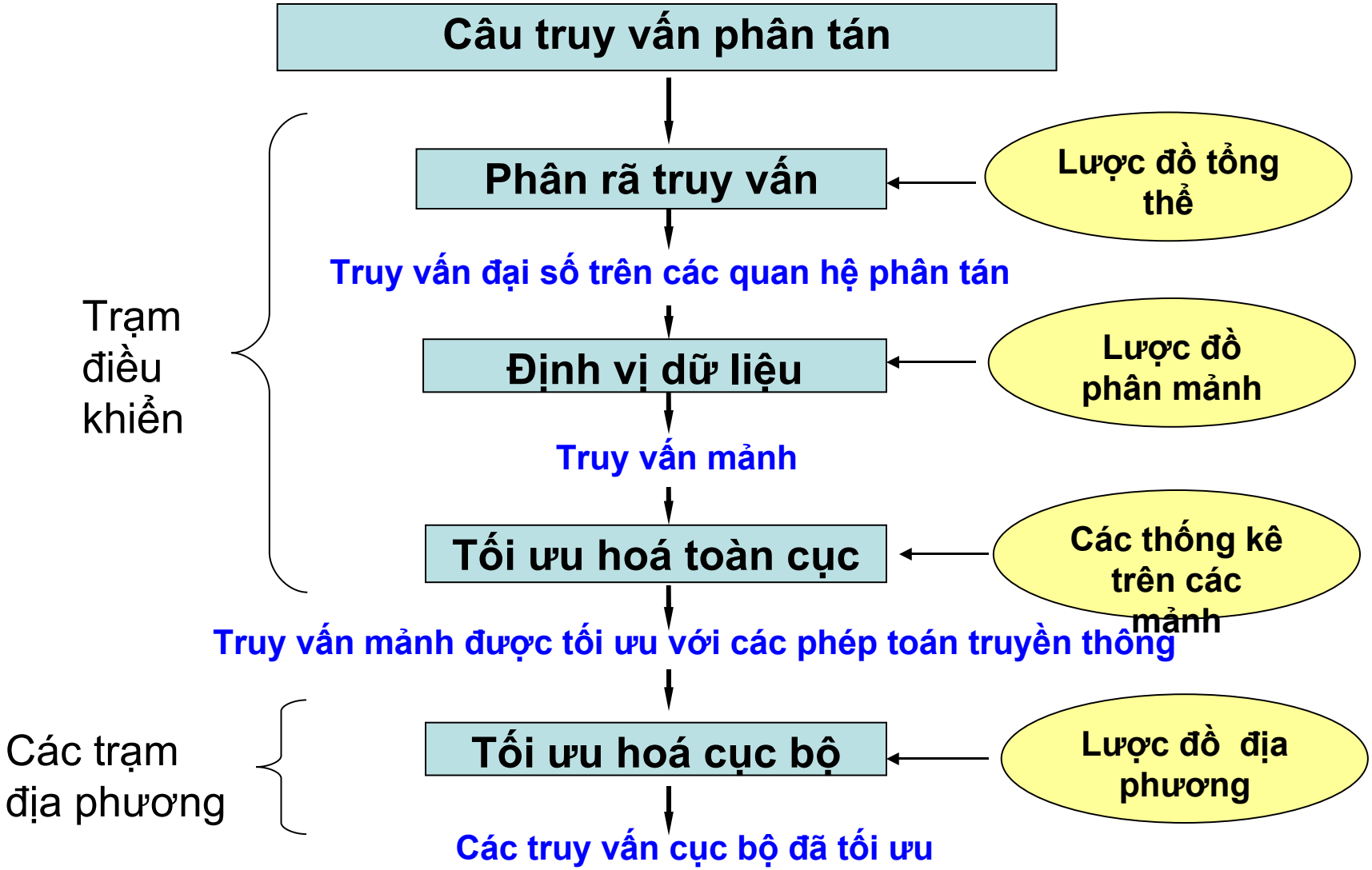
- ***Tập trung:***
  - Chọn một truy vấn đại số quan hệ tốt nhất trong số tất cả các truy vấn đại số tương đương.
  - Các chiến lược xử lý truy vấn có thể biểu diễn trong sự mở rộng của đại số quan hệ.
- ***Phân tán***
  - Kế thừa chiến lược xử lý truy vấn như môi trường tập trung
  - Còn phải quan tâm thêm
    - Các phép toán truyền dữ liệu giữa các trạm
    - Chọn các trạm tốt nhất để xử lý dữ liệu
    - Cách thức và biến đổi dữ liệu

Trong m«i tr-êng tËp trung

Sơ đồ chung



Trong m«i tr-êng phân tán



Sơ đồ phân lớp chung cho xử lý truy vấn phân tán

4.4.2 Chiến lược tối ưu trong CSDL tập trung

*Hai trong số những kỹ thuật tối ưu thông dụng nhất trong các hệ thống tập trung là các thuật toán **INGRES** và **SYSTEM R**.*

Tại sao phải nghiên cứu xử lý truy vấn tập trung?

Để hiểu được các kỹ thuật tối ưu phân tán vì ba lí do:

- Thứ nhất, câu truy vấn phân tán phải được dịch thành các câu truy vấn cục bộ, và được xử lí theo phương pháp tập trung.
- Thứ hai, các kỹ thuật tối ưu hoá phân tán thường là các mở rộng của kỹ thuật tập trung.
- Cuối cùng, tối ưu hoá tập trung thường đơn giản.

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Thuật toán INGRES

***Ý tưởng thuật toán**:* Thuật toán tổ hợp hai giai đoạn phân rã và tối ưu hoá.

- Đầu tiên phân rã câu truy vấn dạng phép toán quan hệ thành các phần nhỏ hơn. Câu truy vấn được phân rã thành một chuỗi các truy vấn có một quan hệ chung duy nhất
- Sau đó mỗi câu truy vấn đơn quan hệ được xử lí bởi một "*thể xử lý truy vấn một biến*" (one variable query processor-OVQP)

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

OVQP tối ưu hoá việc truy xuất đến một quan hệ bằng cách dựa trên vị từ phương pháp truy xuất hữu hiệu nhất đến quan hệ đó

Trước tiên OVQP sẽ thực hiện các phép toán đơn ngôi và giảm thiểu kích thước của các kết quả trung gian bằng các *tách* (detachment) và *thay thế* (substitution)

Kí hiệu $q_{i-1} \rightarrow q_i$ để chỉ câu truy vấn q được phân rã thành hai câu truy vấn con q_{i-1} và q_i, trong đó q_{i-1} được thực hiện trước và kết quả sẽ được q_i sử dụng.

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Phép tách: OVQP sử dụng để tách câu truy vấn q thành các truy vấn q'→q" dựa trên một quan hệ chung là kết quả của q'.

Nếu câu truy vấn q được biểu diễn bằng SQL có dạng:

q: **SELECT** $R_2.A_2, R_3.A_3, \ldots, R_n.A_n$
FROM $R_1, R_2, \ldots, R_n$
WHERE $P_1(R_1.A'_1)$ **AND** $P_2(R_1.A_1, R_2.A_2, \ldots, R_n.A_n)$

Trong đó: A_1 và A'_1 là các thuộc tính của quan hệ R_1,

P_1 là vị từ có chứa các thuộc tính của các quan hệ $R_1, R_2, \ldots, R_n$. Một câu truy vấn như thế có thể phân rã thành hai câu truy vấn con, q' theo sau là q" qua phép tách dựa trên quan hệ chung R_1 như sau:

q': **SELECT** R_1A_1 **INTO** R'_1
FROM R_1
WHERE $P_1(R_1.A_1)$

Trong đó R'_1 là một quan hệ tạm thời chứa các thông tin cần thiết để thực hiện tiếp tục câu truy vấn:

q":**SELECT** $R_2A_2, \ldots, R_nA_n$
FROM $R'_1, R_2, \ldots, R_n$

Ví dụ minh họa: xét CSDL của một công ty máy tính

NHANVIEN (E)

MANV	TENNV	CHUCVU
A1	Nam	Phân tích HT
A2	Trung	Lập trình viên
A3	Đông	Phân tích HT
A4	Bắc	Phân tích HT
A5	Tây	Lập trình viên
A6	Hùng	Kỹ sư điện
A7	Dũng	Phân tích HT
A8	Chiến	Thiết kế DL

HOSO (G)

MANV	MADA	NHIEMVU	THOIGIAN
A1	D1	Quản lý	12
A2	D1	Phân tích	34
A2	D2	Phân tích	6
A3	D3	Kỹ thuật	12
A3	D4	Lập trình	10
A4	D2	Quản lý	6
A5	D2	Quản lý	20
A6	D4	Kỹ thuật	36
A7	D3	Quản lý	48
A8	D3	Lập trình	15

DUAN (J)

MADA	TENDA	NGANSACH
D1	CSDL	20000
D2	CÀI ĐẶT	12000
D3	BẢO TRÌ	28000
D4	PHÁT TRIỂN	25000

TLUONG (S)

CHUCVU	LUONG
Kỹ sư điện	1000
Phân tích HT	2500
Lập trình viên	3000
Thiết kế DL	4000

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Để minh hoạ kỹ thuật tách chúng ta sử dụng CSDL trên cho câu truy vấn sau:"***Cho biết tên của các nhân viên đang làm việc trong dự án có tên CSDL***" Câu truy vấn này (q_1) được diễn tả bằng SQL:

```
q1:   SELECT   NHANVIEN.TENNV
      FROM     NHANVIEN, HOSO, DUAN
      WHERE    NHANVIEN.MANV = HOSO.MANV
               AND  HOSO.MADA = DUAN.MADA
               AND  TENDA = "CSDL"
```

Câu truy vấn q_1 được tách thành $q_{11} \rightarrow q'$, trong đó TGIAN1 là quan hệ trung gian.

```
q11:  SELECT   DUAN.MADA  INTO  TGIAN1
      FROM     DUAN
      WHERE    TENDA = "CSDL"

q':   SELECT   NHANVIEN.TENNV
      FROM     NHANVIEN, HOSO, TGIAN1
      WHERE    NHANVIEN.MANV = HOSO.MANV
               AND HOSO.MADA =TGIAN1.MADA
```

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Các bước tách tiếp theo cho q' có thể tạo ra:

q_{12}: **SELECT** HOSO.MANV INTO TGIAN2
FROM HOSO, TGIAN1
WHERE HOSO.MADA =TGIAN1.MADA

q_{13}: **SELECT** NHANVIEN.TENNV
FROM NHANVIEN, TGIAN2
WHERE NHANVIEN.MANV = TGIAN2.MANV

Truy vấn q_1 đã được rút gọn thành chuỗi truy vấn $q_{11} \rightarrow q_{12} \rightarrow q_{13}$. Truy vấn q_{11} là loại đơn quan hệ và có thể cho thực hiện bởi OVQP. Tuy nhiên các truy vấn q_{12} và q_{13} không phải loại đơn quan hệ và cũng không thể rút gọn hơn nữa bằng phép tách.

Các câu truy vấn đa quan hệ không thể tách tiếp được nữa (chẳng hạn q_{12} và q_{13}) được gọi là bất khả giản (irreducible).

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Các truy vấn bất khả giản được biến đổi thành câu truy vấn đơn quan hệ nhờ *phép thế bộ* (tuple substitution).

Cho câu truy vấn n-quan hệ q, các bộ của một biến được thay bằng các giá trị của chúng, tạo ra được một tập các truy vấn (n-1) biến.

Phép thế bộ được tiến hành như sau:

Trước tiên chọn một quan hệ trong truy vấn q để thay thế. Gọi R_1 là quan hệ đó. Thế thì với mỗi bộ t_{1i} trong R_1, các thuộc tính được tham chiếu trong q được thay bằng các giá trị thật sự trong t_{1i}, tạo ra một câu truy vấn q' có (n-1) quan hệ. Vì vậy số câu truy vấn q' được sinh ra bởi phép thế bộ là card(R_1).

Phép thế bộ có thể tóm tắt như sau:

$q(R_1, R_2, \ldots, R_n)$ được thay bởi $\{q'(t_{1i}, R_2, R_3, \ldots, R_n), t_{1i} \in R_1\}$

Vì thế đối với mỗi bộ thu được, câu truy vấn con được xử lý đệ quy bằng phép thế nếu nó chưa bất khả giản.

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Ví dụ minh họa:

Xét tiếp câu truy vấn q_{13}

q_{13}: **SELECT** NHANVIEN.TENNV
FROM NHANVIEN, TGIAN2
WHERE NHANVIEN.MANV = TGIAN2.MANV

Quan hệ được định nghĩa bởi biến TGIAN2 chạy trên thuộc tính duy nhất MANV. Giả sử rằng nó chỉ chứa hai bộ: <E1> và <E2>. Phép thế cho TGIAN2 tạo ra hai câu truy vấn con đơn quan hệ:

q_{131}: **SELECT** NHANVIEN.TENNV
FROM NHANVIEN
WHERE NHANVIEN.MANV = “E1”

q_{132}: **SELECT** NHANVIEN.TENNV
FROM NHANVIEN
WHERE NHANVIEN.MANV = “E2”

Sau đó chúng có thể được OVQP quản lý và sử dụng.

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Nhận xét:

•Thuật toán tối ưu hoá INGRES (được gọi là INGRES - QOA) sẽ xử lý đệ qui cho đến khi không còn câu truy vấn đa quan hệ nào nữa.

•Thuật toán có thể được áp dụng cho các phép chọn và các phép chiếu ngay khi có thể sử dụng kỹ thuật tách.

•Kết quả của câu truy vấn đơn quan hệ được lưu trong những cấu trúc dữ liệu có khả năng tối ưu hoá những câu truy vấn sau đó (như các nối) và sẽ được OVQP sử dụng.

•Các câu truy vấn bất khả giản còn lại sau phép tách sẽ được sử lý bằng phép thế bộ.

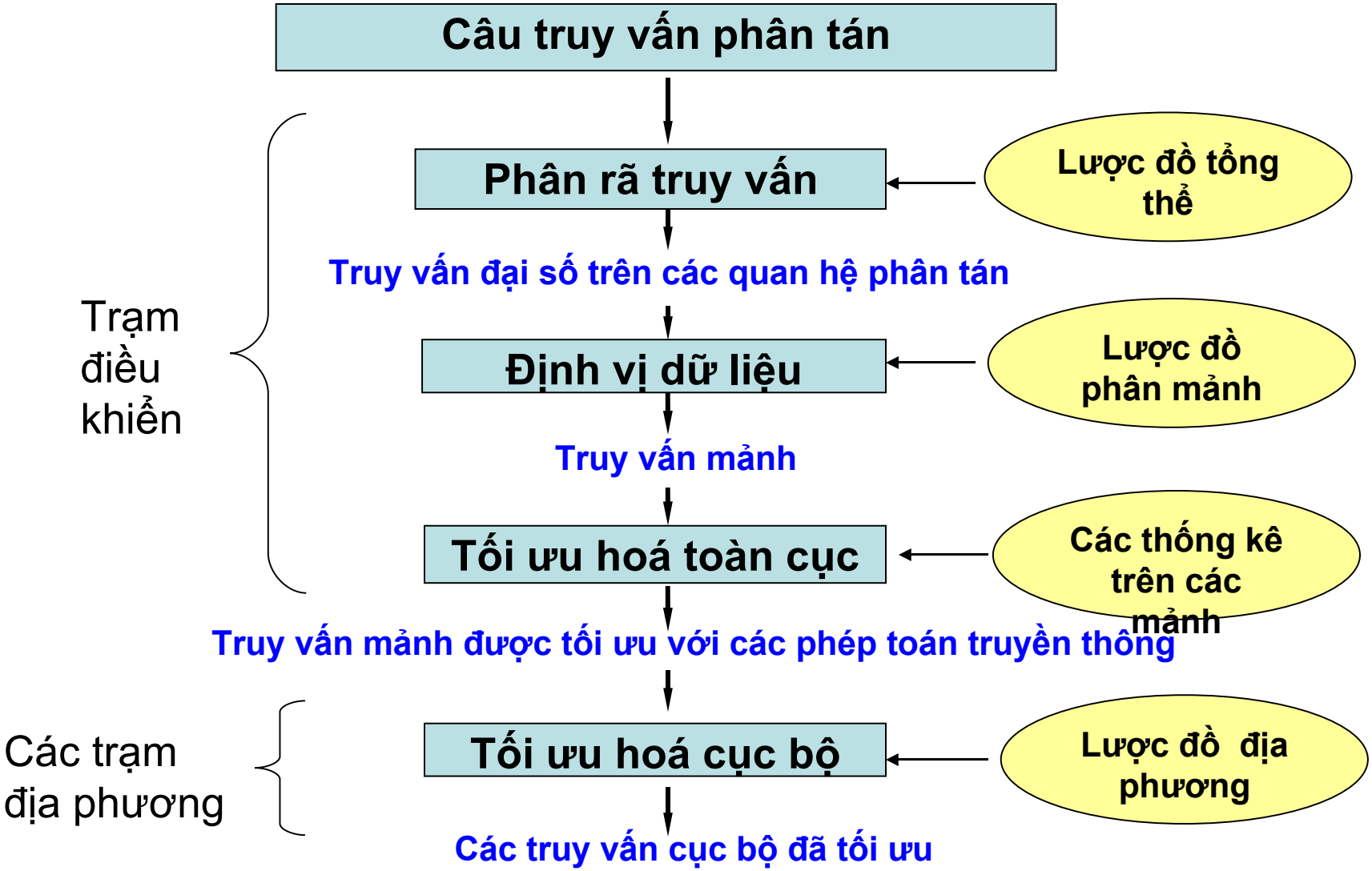
•Câu truy vấn bất khả giản, được kí hiệu là MRQ'. Quan hệ nhỏ nhất với lực lượng của nó đã được biết từ kết quả của câu truy vấn trước đó sẽ được chọn để thay thế.

4.2 XỬ LÝ TRUY VẤN TRONG MÔI TRƯỜNG TẬP TRUNG

Thuật toán INGRES- QOA

Input: MRQ: câu truy vấn đa quan hệ (có n quan hệ)
Output: Câu truy vấn tối ưu
Begin
Output $\leftarrow \phi$
If n=1 **then**
Output $\leftarrow$ run(MRQ) {thực hiện câu truy vấn một quan hệ}
Else {Tách MRQ thành m tr.vấn một quan hệ và một tr.vấn đa quan hệ}
ORQ_1, ..., ORQ_m, MRQ'$\leftarrow$ MRQ
For i$\leftarrow$1 **to** m
Output' $\leftarrow$ run(ORQ_i) {thực hiện ORQ_i }
Output $\leftarrow$ output $\cup$ output' {trộn tất cả các kết quả lại}
Endfor
R $\leftarrow$ CHOOSE_ VARIABLE(MRQ') {R được chọn cho phép thế bộ}
For mỗi bộ t $\in$ R
MRQ" $\leftarrow$ thay giá trị cho t trong MRQ'
Output' $\leftarrow$ INGRES-QOA(MRQ") {gọi đệ qui}
Output $\leftarrow$ output $\cup$ output' {trộn tất cả các kết quả lại}
Endfor
Endif
End. {INGRES-----QOA}

4.3 Xử lý truy vấn trong môi trường phân tán



Sơ đồ phân lớp chung cho xử lý truy vấn phân tán

4.3.1 Phân rã truy vấn

- Biến đổi một phép tính quan hệ thành một truy vấn đại số trên quan hệ tổng thể.
- Cả hai truy vấn vào/ra đều được thực hiện trên quan hệ tổng thể và không quan tâm đến tính phân tán của dữ liệu.
- Vì vậy, phân rã truy vấn được thực hiện chung cho cả hệ tập trung và phân tán.
- Trong phần này chúng ta giả sử rằng các truy vấn vào luôn cú pháp đúng. Khi giai đoạn xử lý truy vấn thực hiện xong, thì truy vấn ra là đúng và tránh được các công việc dư thừa.
- Giai đoạn này chia làm bốn bước: chuẩn hoá, phân tích, loại bỏ dư thừa và viết lại.

Chúng ta trình bày ba bước đầu tiên trong phạm vi của phép tính quan hệ bộ. Chỉ có bước cuối cùng ghi truy vấn lại thành đại số quan hệ.

4.3.1.1 Chuẩn hoá

Mục đích: chuyển đổi truy vấn thành một dạng chuẩn để thuận lợi cho các xử lý tiếp theo.

Với SQL, có hai dạng chuẩn cho các tân từ trong mệnh đề WHERE là:

Dạng chuẩn hội là hội ($\wedge$) của những phép toán tuyển ($\vee$):

$(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$

Dạng chuẩn tuyển là tuyển ($\vee$) của những phép toán hội ($\wedge$):

$(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$, trong đó p_{ij} là các biểu thức nguyên tố.

4.3 Xử lý truy vấn trong môi trường phân tán

Các quy tắc biến đổi tương đương trên các phép toán logic:

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $\neg (\neg p) \Leftrightarrow p$
4. $(p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$
5. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$.
6. $p_1 \wedge (p_2 \wedge p_3) \Leftrightarrow (p_1 \wedge p_2) \wedge p_3$
7. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
8. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
9. $(p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$

4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ:

Từ các quan hệ NHANVIEN (MANV, TENNV, CHUCVU) và HOSO (MANV, MADA, NHIEMVU, THOIGIAN). Xét truy vấn:

"*Tìm tên các nhân viên làm dự án J1 có thời gian 12 hoặc 24 tháng*" .

Truy vấn trên được biểu diễn trong SQL:

```
SELECT      NHANVIEN. TENNV
FROM        NHANVIEN, HOSO
WHERE       NHANVIEN.MANV= HOSO.MANV
      AND   HOSO.MADA="J1"
      AND   THOIGIAN=12  OR  THOIGIAN=24
```

Điều kiện trong dạng chuẩn hội là:

NHANVIEN.MANV=HOSO.MANV $\wedge$ HOSO.MADA="J1" $\wedge$ (THOIGIAN=12 $\vee$ THOIGIAN=24)

Điều kiện trong dạng chuẩn tuyển là:

(NHANVIEN.MANV=HOSO.MANV $\wedge$ HOSO.MADA="J1" $\wedge$THOIGIAN=12)
$\vee$ (NHANVIEN.MANV=HOSO.MANV $\wedge$ HOSO.MADA="J1" $\wedge$ THOIGIAN=24)

4.3.1.2 Phân tích

Mục đích: Phát hiện ra những thành phần không đúng (sai kiểu hoặc sai ngữ nghĩa) và loại bỏ chúng sớm nhất nếu có thể.

Truy vấn sai kiểu: nếu một thuộc tính bất kỳ hoặc tên quan hệ của nó không được định nghĩa trong lược đồ tổng thể, hoặc phép toán áp dụng cho các thuộc tính sai kiểu.

Ví dụ: truy vấn dưới đây là sai kiểu

```
SELECT    E#
FROM      E
WHERE     E.TENNV > 200
```

vì hai lý do:

- Thuộc tính E# không khai báo trong lược đồ
- Phép toán “>200” không thích hợp với kiểu chuỗi của thuộc tính E.TENNV

4.3 Xử lý truy vấn trong môi trường phân tán

Truy vấn sai ngữ nghĩa: nếu các thành phần của nó không tham gia vào việc tạo ra kết quả.

Để xác định truy vấn có sai về ngữ nghĩa hay không, ta dựa trên việc biểu diễn truy vấn như một đồ thị gọi là *đồ thị truy vấn*. Đồ thị này được xác định bởi các truy vấn liên quan đến phép chọn, chiếu và nối. Nếu đồ thị truy vấn mà không liên thông thì truy vấn là sai ngữ nghĩa

4.3 Xử lý truy vấn trong môi trường phân tán

Đồ thị truy vấn:

- Một nút dùng để biểu diễn cho quan hệ kết quả
- Các nút khác biểu diễn cho các toán hạng trong quan hệ
- Cạnh nối giữa hai nút không phải là nút kết quả biểu diễn một phép nối.
- Cạnh có nút đích là kết quả thì biểu diễn một phép chiếu.
- Một nút không phải là kết quả có thể được gán nhãn bởi phép chọn hoặc phép tự nối (seft-join: nối của quan hệ với chính nó).

Đồ thị kết nối:

- Là một đồ thị con của đồ thị truy vấn (join graph), trong đó chỉ có phép nối.

4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ: Từ các quan hệ E=NHANVIEN (MANV, TENNV, CHUCVU) và G = HOSO (MANV, MADA, NHIEMVU, THOIGIAN) và J=DUAN (MADA, TENDA, NGANSACH).

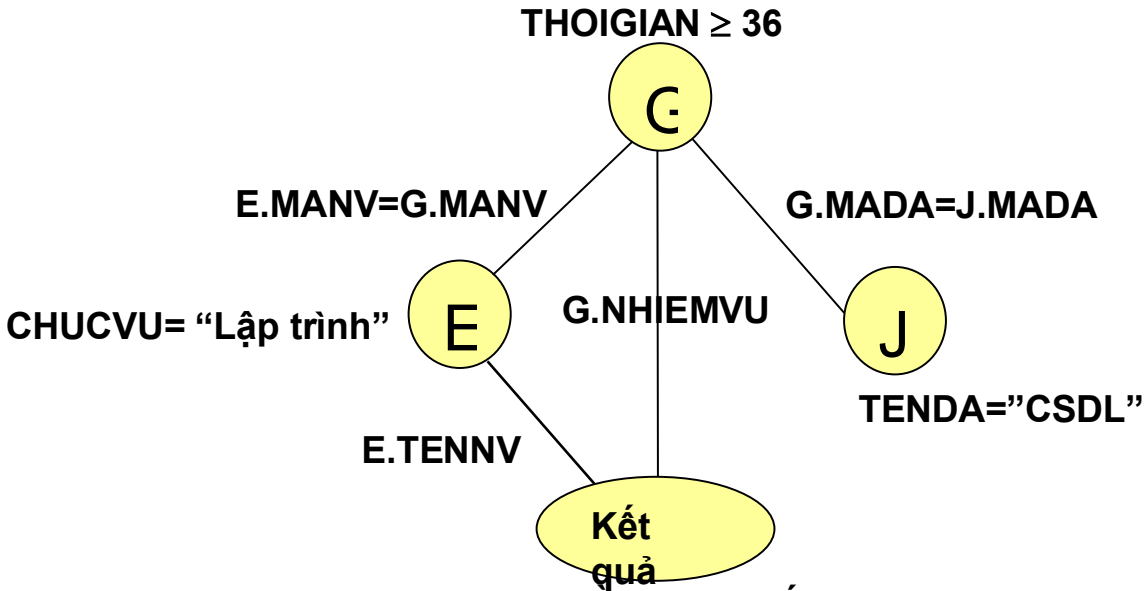
Hãy xác định "***Tên và nhiệm vụ các lập trình viên làm dự án CSDL có thời gian lớn hơn 3 năm.***"

Truy vấn SQL tương ứng là:

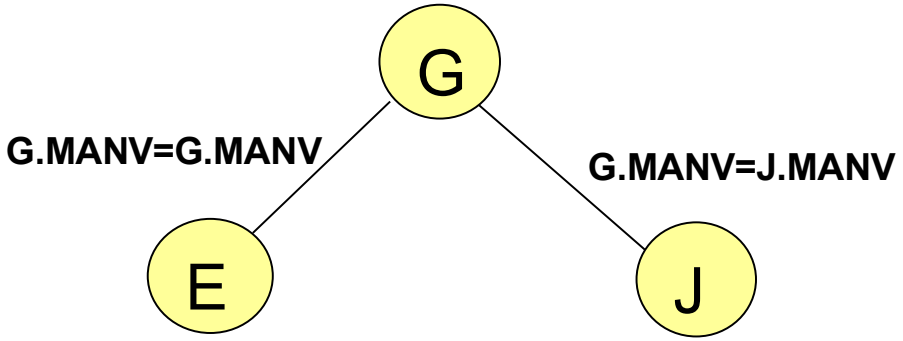
```
SELECT    E.TENNV, G.NHIEMVU
FROM      E, G, J
WHERE     E.MANV=G.MANV
          AND   G.MADA.= J.MADA
          AND   TENDA="CSDL"
          AND   THOIGIAN≥ 36
          AND   CHUCVU="LTRINH"
```

4.3 Xử lý truy vấn trong môi trường phân tán

Đồ thị truy vấn và đồ thị kết nối tương ứng



(a) Đồ thị truy vấn



(b) Đồ thị kết nối tương ứng

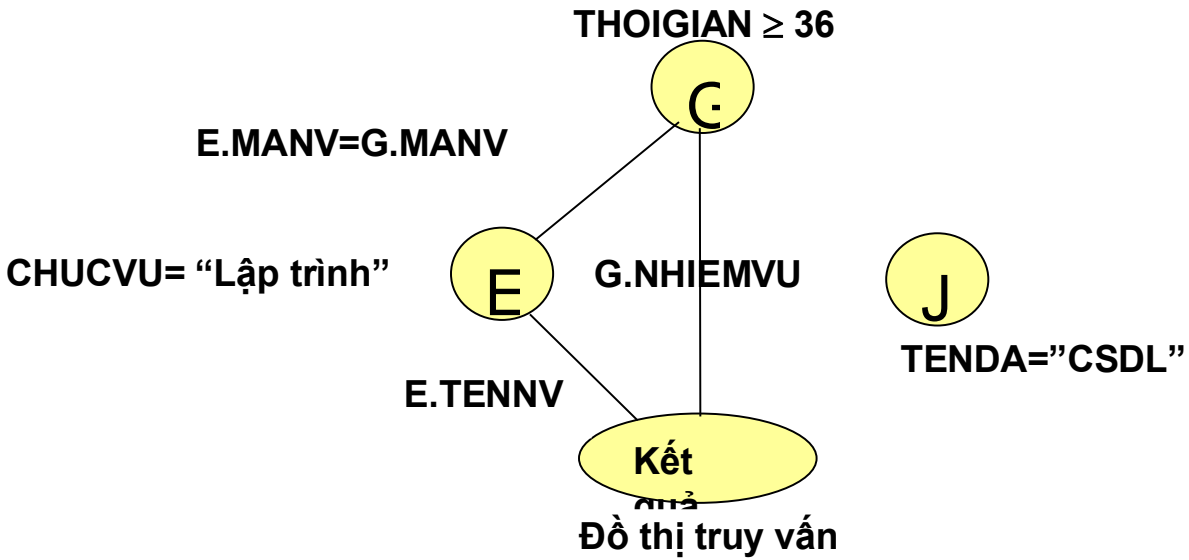
4.3 Xử lý truy vấn trong môi trường phân tán

Câu truy vấn SQL tương ứng:

thiếu AND G.MADA=J.MADA

```
SELECT   E.TENNV,  NHIEMVU
FROM     E, G, J
WHERE    E.MANV=G.MANV
         AND   TENDA=”CSDL”
         AND   THOIGIAN ≥ 36
         AND   CHUCVU=”Lập trình”
```

Truy vấn này là sai ngữ nghĩa vì đồ thị truy vấn của nó không liên thông.



Đồ thị truy vấn

4.3.1.3 Loại bỏ dư thừa

- Điều kiện trong các truy vấn có thể có chứa các tân từ dư thừa.
- Một đánh giá sơ sài về một điều kiện dư thừa có thể dẫn đến lặp lại một số công việc.
- Sự dư thừa tân từ và dư thừa công việc có thể được loại bỏ bằng cách làm đơn giản hoá các điều kiện thông qua các luật luỹ đẳng sau:

1. $p \wedge p \Leftrightarrow p$
2. $p \vee true \Leftrightarrow true$
3. $p \vee p \Leftrightarrow p$
4. $p \wedge \neg p \Leftrightarrow false$
5. $p \wedge true \Leftrightarrow p$
6. $p \vee \neg p \Leftrightarrow true$
7. $p \vee false \Leftrightarrow p$
8. $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
9. $p \wedge false \Leftrightarrow false$
10. $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

Ví dụ: xét câu truy vấn sau:

4.3 Xử lý truy vấn trong môi trường phân tán

```
SELECT   G.CHUCVU
FROM     E
WHERE    (NOT(G.CHUCVU=”Lập trình”)
         AND (G.CHUCVU=”Lập trình”  OR G.CHUCVU=”Kỹ sư điện”)
         AND NOT(G.CHUCVU=”Kỹ sư điện”)
         OR   E.TENNV=”Dung”
```

Sử dụng các luật lũy đẳng nêu trên, truy vấn được biến đổi thành:

```
SELECT        G.CHUCVU
FROM          E
WHERE         E.TENNV=”Dung”
```

Thực vậy, đặt p1:<CHUCVU=”Lập trình”>, p2:<CHUCVU=”Kỹ sư điện”>, p3: <E.TENNV=”Dung”>.

Khi đó, các tân từ sau mệnh đề WHERE được mô tả lại:

p: $(\neg p1 \wedge (p1 \vee p2) \wedge \neg p2) \vee p3$

$\Leftrightarrow (\neg p1 \wedge p1 \wedge \neg p2) \vee (\neg p1 \wedge p2 \wedge \neg p2) \vee p3$ (áp dụng luật 7)

$\Leftrightarrow (false \wedge \neg p2) \vee (\neg p1 \wedge false) \vee p3$ (áp dụng luật 5)

$\Leftrightarrow false \vee false \vee p3$ (áp dụng luật 4)

$\Leftrightarrow p3$

4.3.1.4 Viết lại

Bước này được chia làm hai bước con như sau:

- Biến đổi trực tiếp truy vấn phép tính sang đại số quan hệ.
- Cấu trúc lại truy vấn đại số quan hệ để cải thiện hiệu quả thực hiện.

Thông thường người ta biểu diễn các truy vấn đại số quan hệ bởi *cây đại số quan hệ*.

Cây đại số quan hệ là một cây mà nút lá biểu diễn một quan hệ trong CSDL, các nút không lá là các quan hệ trung gian được sinh ra bởi các phép toán đại số quan hệ.

4.3 Xử lý truy vấn trong môi trường phân tán

Cách chuyển một truy vấn phép tính quan hệ thành một cây đại số quan hệ:

- Các nút lá khác nhau được tạo cho mỗi biến bộ khác nhau (tương ứng một quan hệ). Trong SQL các nút lá chính là các quan hệ trong mệnh đề FROM.
- Nút gốc được tạo ra xem bởi một phép chiếu lên các thuộc tính kết quả. Trong SQL nút gốc được xác định qua mệnh đề SELECT.
- Điều kiện (mệnh đề WHERE trong SQL) được biến đổi thành dãy các phép toán đại số thích hợp (phép chọn, nối, phép hợp, v.v...) đi từ lá đến gốc, có thể thực hiện theo thứ tự xuất hiện của các tân từ và các phép toán.

4.3 Xử lý truy vấn trong môi trường phân tán

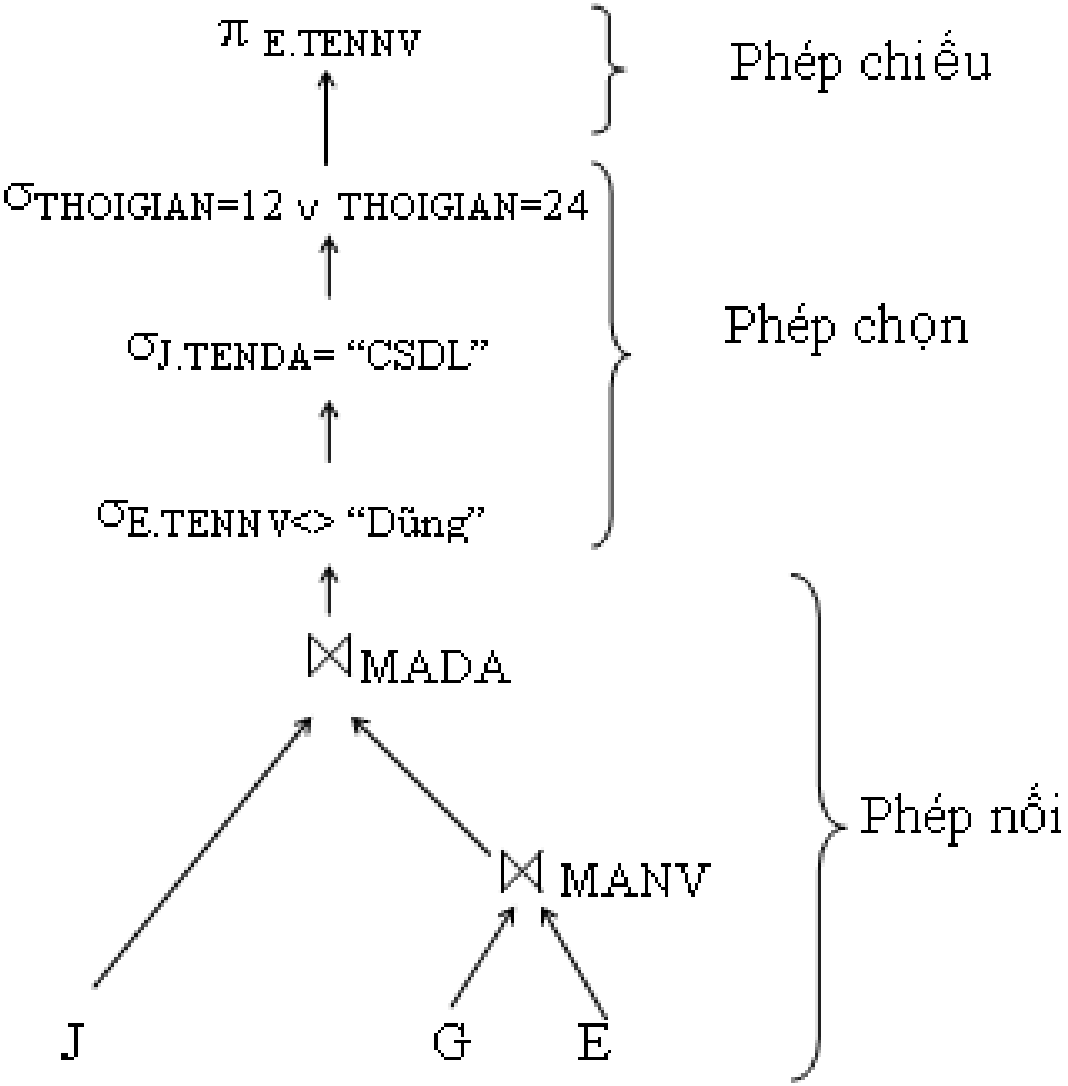
Ví dụ:

Truy vấn *"Tìm tên các nhân viên không phải là "Dũng", làm việc cho dự án CSDL với thời gian một hoặc hai năm"*.

Biểu diễn truy vấn này trong SQL là:

```
SELECT    TENNV
FROM      J, G, E
WHERE     G.MANV=E.MANV
      AND G.MADA= J.MADA
      AND E.TENNV <> "Dũng"
      AND J.TENDA= "CSDL"
      AND (THOIGIAN=12  OR  THOIGIAN=24)
```

4.3 Xử lý truy vấn trong môi trường phân tán



Hình 4.4: Cây đại số quan hệ

06 luật biến đổi phép toán đại số quan hệ:

Mục đích: dùng để biến đổi cây đại số quan hệ thành các cây tương đương (trong đó có thể có cây tối ưu).

Giả sử R, S, T là các quan hệ, R được định nghĩa trên toàn bộ thuộc tính $A=\{A_1, ..., A_n\}$, S được định nghĩa trên toàn bộ thuộc tính $B=\{B_1, ..., B_n\}$.

1.Tính giao hoán của các phép toán hai ngôi:

Phép tích Decartes và phép nối hai quan hệ có tính giao hoán.

i. $R \times S \Leftrightarrow S \times R$ ii. $R \bowtie S \Leftrightarrow S \bowtie R$

2. Tính kết hợp của các phép toán hai ngôi:

Phép tích Decartes và phép nối hai quan hệ có tính kết hợp.

i. $(R \times S) \times T \Leftrightarrow R \times (S \times T)$ ii. $(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$

3. *Tính luỹ đẳng của những phép toán một ngôi*

- Dãy các phép chiếu khác nhau trên cùng quan hệ được tổ hợp thành một phép chiếu và ngược lại:

$$\Pi_{A'}(\Pi_{A''}(R)) \Leftrightarrow \Pi_{A'}(R) \quad A', A'' \in R \text{ và } A' \subseteq A''$$

- Dãy các phép chọn khác nhau $\sigma_{pi(Ai)}$ trên cùng một quan hệ, với p_i là một tân từ được gán vào thuộc tính A_i , có thể được tổ hợp thành một phép chọn.

$$\sigma_{p1(A1)}(\sigma_{p2(A2)}(R)) = \sigma_{p1(A1) \wedge p2(A2)}(R)$$

4. Phép chọn giao hoán với phép chiếu

$$\Pi_{A1,..., An}(\sigma_{p\,(Ap)}(R)) \Leftrightarrow \Pi_{A1,.., An}(\sigma_{p(Ap)}(\Pi_{A1,...,An,Ap}(R))$$

Nếu Ap là thành viên của {A1, ..., An}, biểu thức trên thành

$$\Pi_{A1, ..., An}(\sigma_{p(Ap)}(R)) \Leftrightarrow \sigma_{p(Ap)}(\Pi_{A1, ..., An}(R))$$

5. Phép chọn giao hoán với những phép toán hai ngôi

- Phép chọn với phép nhân: $\sigma_{p(Ai)}(R \times S) \Leftrightarrow \sigma_{p(Ai)}(R) \times S$

- Phép chọn với phép nối:

  $\sigma_{p(Ai)}(R \bowtie_{p(Ai,Bk)} S) \Leftrightarrow \sigma_{p(Ai)}(R) \bowtie_{(Aj,Bk)} S$

- Phép chọn với phép hợp: Nếu R và T cùng bộ thuộc tính.

  $\sigma_{p(Ai)}(R \cup T) \Leftrightarrow \sigma_{p(Ai)}(R) \cup \sigma_{p(Ai)}(T)$

6. Phép chiếu giao hoán với những phép toán hai ngôi

- ***Phép chiếu và tích Decartes***: Nếu C=A'∪B' với A'⊆ A, B'⊆ B, và A, B là tập các thuộc tính trên quan hệ R, S ta có:

  $$\Pi_C(R \times S) = \Pi_{A'}(R) \times \Pi_{B'}(S)$$

- ***Phép chiếu và phép nối***:

  $$\Pi_C(R \bowtie_{p(Ai,Bj)} S) = \Pi_{A'}(R) \bowtie_{p(Ai,Bj)} \Pi_{B'}(S)$$

- ***Phép chiếu và phép hợp:***

  $$\Pi_C(R \cup S) = \Pi_{A'}(R) \cup \Pi_{B'}(S)$$

Chú ý: Việc sử dụng sáu luật trên có khả năng sinh ra nhiều cây đại số quan hệ tương đương nhau. Vấn đề là xác định cho được cây tối ưu

4.3 Xử lý truy vấn trong môi trường phân tán

Chú ý:

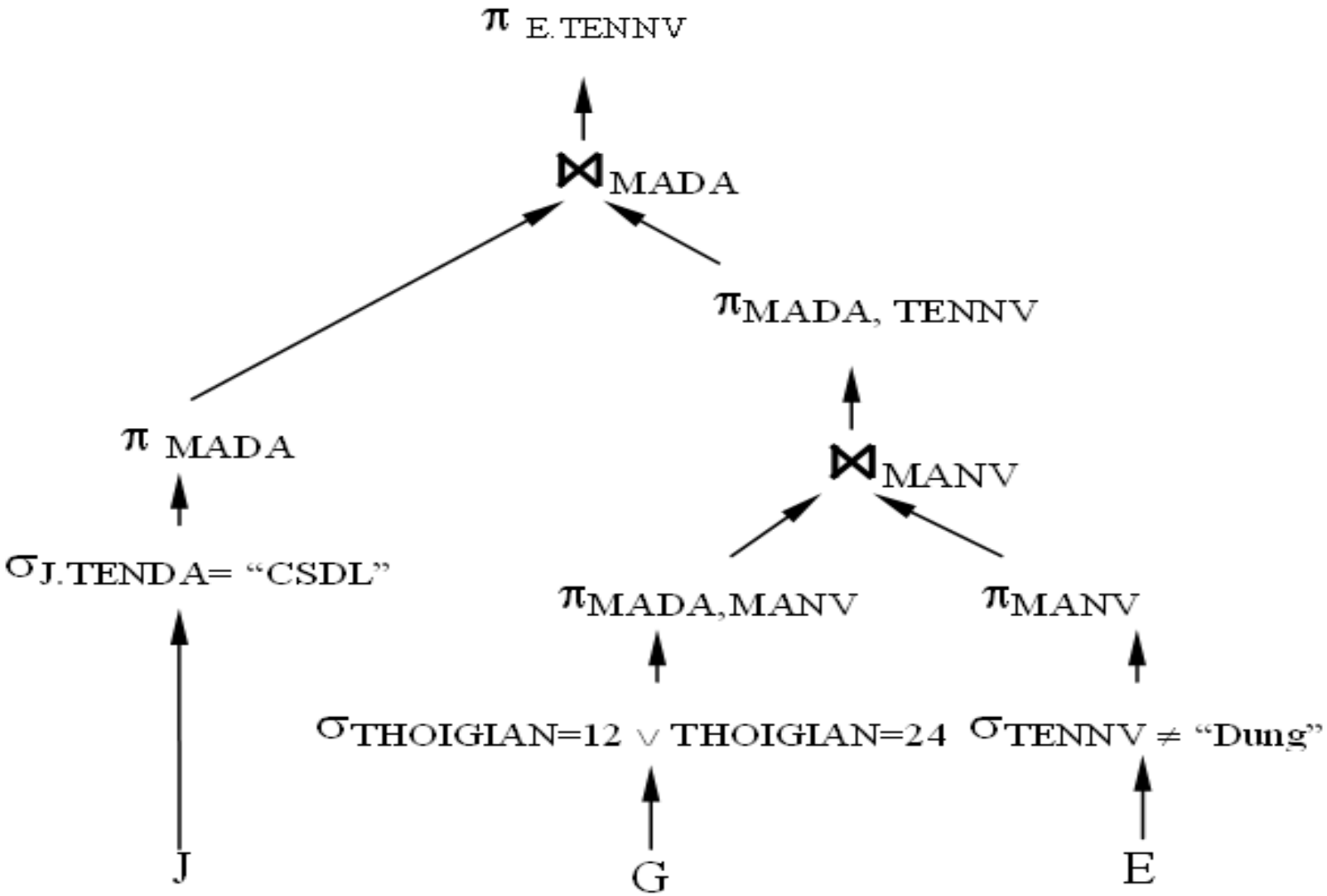
Trong giai đoạn tối ưu, sự so sánh các cây có thể thực hiện dựa trên chi phí dự đoán của chúng. Tuy nhiên, nếu số lượng các cây quá lớn thì cách tiếp cận này sẽ không hiệu quả. Có thể dùng các luật trên để cấu trúc lại cây, nhằm loại bỏ những cây đại số quan hệ "tồi".

Các luật trên có thể sử dụng theo bốn cách như sau:

- Phân rã các phép toán một ngôi, đơn giản hóa biểu thức truy vấn .
- Nhóm các phép toán một ngôi trên cùng một quan hệ để giảm số lần thực hiện.
- Giao hoán các phép toán một ngôi với các phép toán hai ngôi để ưu tiên cho một số phép toán (chẳng hạn phép chọn).
- Sắp thứ tự các phép toán hai ngôi trong thực hiện truy vấn.

4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ: Cấu trúc lại cây truy vấn ở ví dụ trên, cho ra cây kết quả tốt hơn cây ban đầu, tuy nhiên vẫn còn xa cây tối ưu.



4.3.2 Định vị dữ liệu phân tán-Tối ưu hóa cục bộ

- Lớp định vị biến đổi một truy vấn đại số quan hệ tổng thể thành một truy vấn đại số được biểu thị trên các mảnh vật lý.
- Sử dụng thông tin được lưu trữ trên các lược đồ phân mảnh để định vị.
- Chương trình đại số quan hệ xây dựng lại quan hệ tổng thể từ các phân mảnh của nó gọi là *chương trình định vị*.
- Truy vấn có được từ chương trình định vị gọi là truy vấn ban đầu.
- Trong phần dưới đây, với *mỗi kiểu phân mảnh* chúng ta sẽ biểu diễn một *kỹ thuật rút gọn* để sinh ra truy vấn được tối ưu và đơn giản hoá.

4.2.2.1 Rút gọn theo phân mảnh ngang nguyên thuỷ

Xét quan hệ E(MANV,TENNV,CHUCVU). Tách quan hệ này thành ba mảnh ngang E_1, E_2 và E_3 như sau:

$E_1=\sigma_{MANV \leq "E3"}(E)$ $E_2=\sigma_{"E3"< MANV \leq "E6"}(E)$ $E_3=\sigma_{MANV > "E6"}(E)$

Chương trình định vị cho một quan hệ E được phân mảnh ngang là hợp của các mảnh E_1, E_2, E_3.

Nghĩa là, $E = E_1 \cup E_2 \cup E_3$. Vì vậy, dạng ban đầu của bất kỳ truy vấn nào được xác định trên E là có được bằng cách thay thế nó bởi $E_1 \cup E_2 \cup E_3$.

Việc rút gọn các truy vấn trên các quan hệ đã được phân mảnh ngang bao gồm việc *xác định câu truy vấn*, sau khi đã cấu trúc lại cây con. Điều này sẽ sinh ra một số quan hệ rỗng, và sẽ *loại bỏ* chúng. Phân mảnh ngang có thể được khai thác để làm đơn giản cả phép chọn và phép nối.

4.3 Xử lý truy vấn trong môi trường phân tán

a. ***Rút gọn với phép chọn***: cho một quan hệ R được phân mảnh ngang thành $R_1, R_2, ..., R_n$ với $R_j = \sigma_{pj}(R)$.

Luật 1: $\sigma_{pj}(R_j) = \phi$ nếu $\forall x \in R : \neg(p_i(x) \wedge p_j(x))$. Trong đó, p_i, p_j là tân từ chọn, x là bộ dữ liệu, p(x) là tân từ p chiếm giữ x.

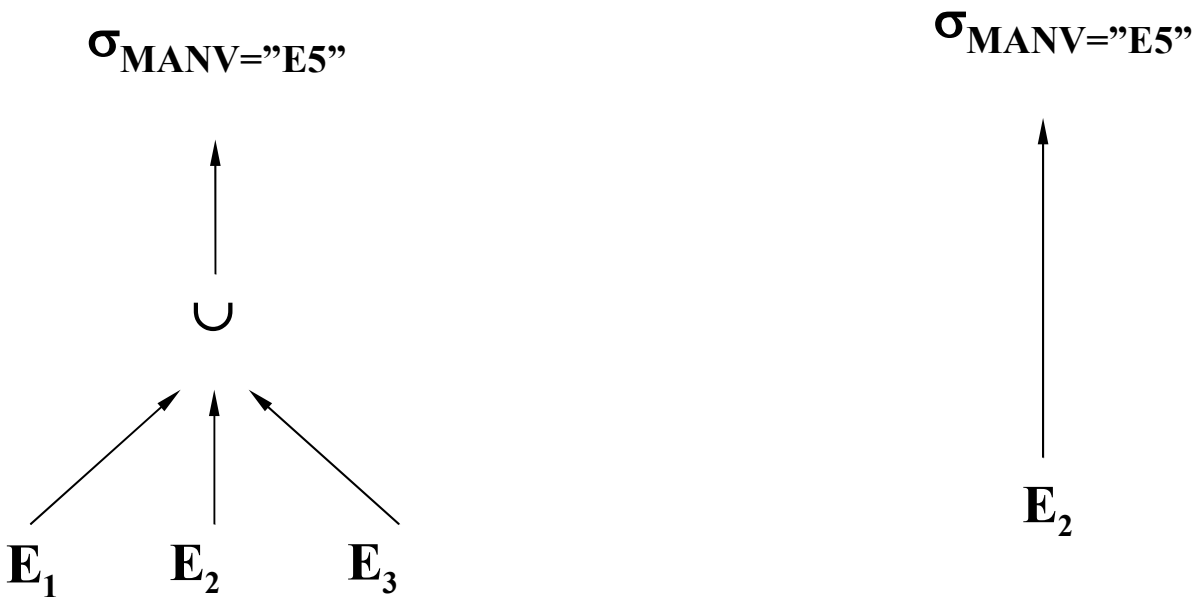
Ví dụ: Xét truy vấn

```
SELECT   *
FROM     E
WHERE    MANV="E5"
```

Áp dụng cách tiếp cận tự nhiên đến vùng E từ E_1, E_2 và E_3 cho truy vấn ban đầu, hình 4.7a. Bằng cách sử dụng tính chất giao hoán phép chọn với phép hợp, chúng ta thấy tân từ chọn đối lập với tân từ E_1 và E_3, sinh ra các quan hệ rỗng. Chúng ta thu được truy vấn rút gọn ở hình 4.7 b.

4.3 Xử lý truy vấn trong môi trường phân tán

$$E_1 = \sigma_{MANV \leq "E3"}(E)$$
$$E_2 = \sigma_{"E3" < MANV \leq "E6"}(E)$$
$$E_3 = \sigma_{MANV > "E6"}(E)$$



(a) Truy vấn ban đầu **(b) Truy vấn rút gọn**

Hình 4.7: Rút gọn cho phân mảnh ngang với phép hợp

4.3 Xử lý truy vấn trong môi trường phân tán

b.Rút gọn với phép nối

- Các phép nối trên quan hệ đã được phân mảnh ngang có thể đơn giản khi chúng được phân mảnh theo thuộc tính nối.
- Việc rút gọn được thực hiện dựa trên tính phân phối giữa phép nối và phép hợp và loại bỏ các phép nối vô ích.

$(R_1 \cup R_2) \bowtie R_3 = (R_1 \bowtie R_3) \cup (R_2 \bowtie R_3)$, R_i là các phân mảnh.

Với tính chất này, có thể xác định được các phép nối vô ích của các mảnh khi các điều kiện nối mâu thuẫn nhau.

Khi đó, dùng luật 2 dưới đây để loại bỏ các phép nối vô ích.

Luật 2: $\mathbf{R_i \bowtie R_j = \phi}$ **nếu** $\mathbf{\forall x \in R_i, \forall y \in R_j : \neg (p_i(x) \wedge p_j(y))}$. Trong đó R_i, R_j được xác định theo các tân từ p_i, p_j trên cùng thuộc tính.

Nhận xét:

- Việc xác định các phép nối vô ích được thực hiện bằng cách chỉ xem xét các tân từ mảnh.
- Truy vấn rút gọn không phải luôn tốt hơn hoặc đơn giản hơn truy vấn ban đầu.
- Một thuận lợi của truy vấn rút gọn là những phép nối có thể thực hiện song song.

4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ: Giả sử quan hệ E được phân mảnh thành các mảnh E_1=

$\sigma_{MANV \leq "E3"}(E)$ $E_2=\sigma_{"E3"< MANV \leq "E6"}(E)$ $E_3=\sigma_{MANV > "E6"}(E)$

Quan hệ G được phân làm hai mảnh:

$G_1=\sigma_{MANV\leq"E3"}(G)$ và $G_2=\sigma_{MANV>"E3"}(G)$.

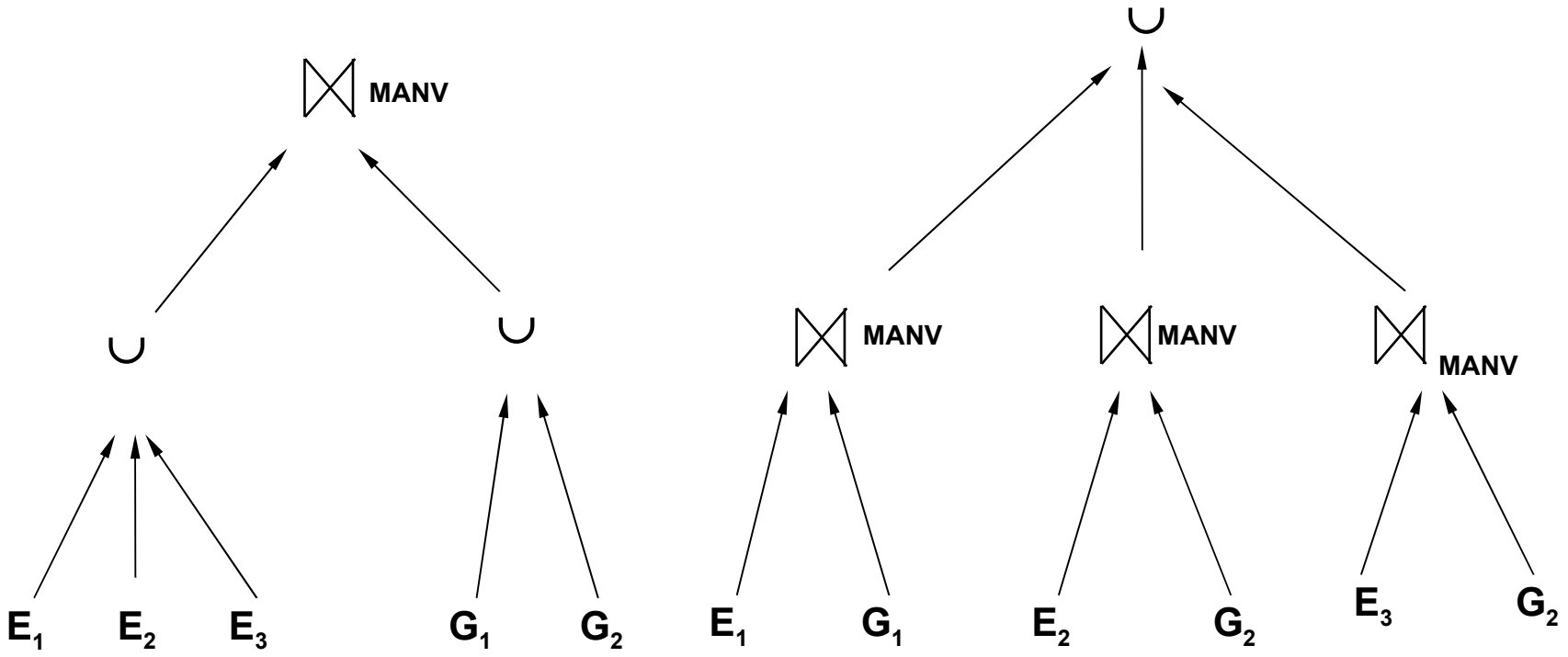
Nhận xét:

- E_1 và G_1 được định nghĩa bởi cùng tân từ.
- Tân từ định nghĩa G_2 là hợp của các định nghĩa của những tân từ E_2 và E_3.

Xét truy vấn

```
SELECT    *
FROM      E, G
WHERE     E.MANV=G.MANV
```

4.3 Xử lý truy vấn trong môi trường phân tán



(a) Truy vấn ban đầu

(b) Truy vấn rút gọn

Hình 4.8: Sự rút gọn phân mảnh ngang với phép nối

4.2.2.2 Rút gọn theo phân mảnh dọc

- Chức năng của việc phân mảnh dọc là tách quan hệ dựa vào thuộc tính của các phép chiếu.
- Vì phép toán xây dựng lại đối với phân mảnh dọc là nối, nên chương trình định vị một quan hệ đã được phân mảnh dọc là nối của các mảnh trong vùng thuộc tính chung.

Ví dụ: Quan hệ E được phân mảnh dọc thành E_1, E_2, với thuộc tính khoá MANV được lặp lại như sau:

$E_1 = \Pi_{MANV,TENNV}(E)$ và $E_2 = \Pi_{MANV,CHUCVU}(E)$

Chương trình định vị là: $E = E_1 \bowtie_{MANV} E_2$

- Các truy vấn trên phân mảnh dọc có thể rút gọn bằng cách xác định những quan hệ trung gian vô ích và loại bỏ các cây con chứa chúng.
- Các phép chiếu trên một phân mảnh dọc không có thuộc tính chung với các thuộc tính chiếu (ngoại trừ khóa của quan hệ) là vô ích, mặc dù các quan hệ là khác rỗng.

4.3 Xử lý truy vấn trong môi trường phân tán

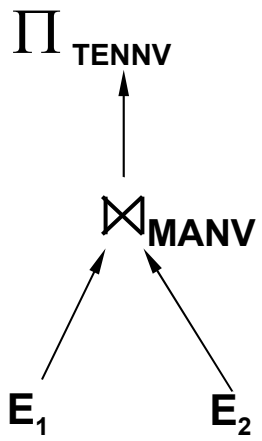
Luật 3: $\Pi_{D,K}(R_i)$ là vô ích nếu $D \cap A' = \phi$. Trong đó, quan hệ R xác định trên $A=\{A_1, ..., A_n\}$; $R = \Pi_{A'}(R)$, $A' \subseteq A$, K là khoá của quan hệ, $K \subset A$, D là tập các thuộc tính chiếu, $D \subset A$.

Ví dụ: Với quan hệ E được phân mảnh dọc như sau:

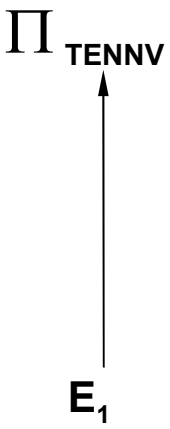
$E_1 = \Pi_{MANV,TENNV}(E)$ và $E_2 = \Pi_{MANV,CHUCVU}(E)$

Xét truy vấn SQL:

```
SELECT    TENNV
FROM      E
```



(a) Truy vấn ban đầu



(b) Truy vấn rút gọn

Hình 4.9: Rút gọn đối với việc phân mảnh dọc

Nhận xét: phép chiếu trên E_2 là vô ích vì TENNV không có trong E_2, nên phép chiếu chỉ cần gán vào E_1

4.2.2.3 Rút gọn theo phân mảnh gián tiếp

- Sự phân mảnh ngang gián tiếp là một cách tách hai quan hệ để việc xử lý nối của các phép chọn và phép nối
- Nếu quan hệ R phụ thuộc vào sự phân mảnh ngang gián tiếp nhờ quan hệ S, thì các mảnh của R và S, mà có cùng giá trị thuộc tính nối, được định vị tại cùng trạm. Ngoài ra, S có thể được phân mảnh tùy thuộc vào tân từ chọn.
- Khi các bộ của R được đặt tuỳ theo những bộ của S, thì sự phân mảnh gián tiếp chỉ nên sử dụng mối quan hệ một nhiều từ S→R (với một bộ của S có thể phù hợp với n bộ của R, Nhưng với một bộ của R chỉ phù hợp với một bộ của S).
- Truy vấn trên các phân mảnh gián tiếp cũng có thể rút gọn được, nếu các tân từ phân mảnh mâu thuẫn nhau thì phép nối sẽ đưa ra quan hệ rỗng.
- Chương trình định vị một quan hệ đã được phân mảnh ngang gián tiếp là hợp của các mảnh.

4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ: Cho mối quan hệ một nhiều từ E đến G, quan hệ G (MANV, MADA, NHIEMVU, THOIGIAN) có thể được phân mảnh gián tiếp theo những luật sau:

$G_1 = G \ltimes_{MANV} E_1$ và $G_2 = G \ltimes_{MANV} E_2$.

Trong đó E được phân mảnh ngang như sau:

$E_1 = \sigma_{CHUCVU=\text{"Lập trình"}}(E)$ và $E_2 = \sigma_{CHUCVU\neq\text{"Lập trình"}}(E)$

Chương trình định vị cho một quan hệ đã được phân mảnh gián tiếp là hợp của các mảnh $G=G_1 \cup G_2$.

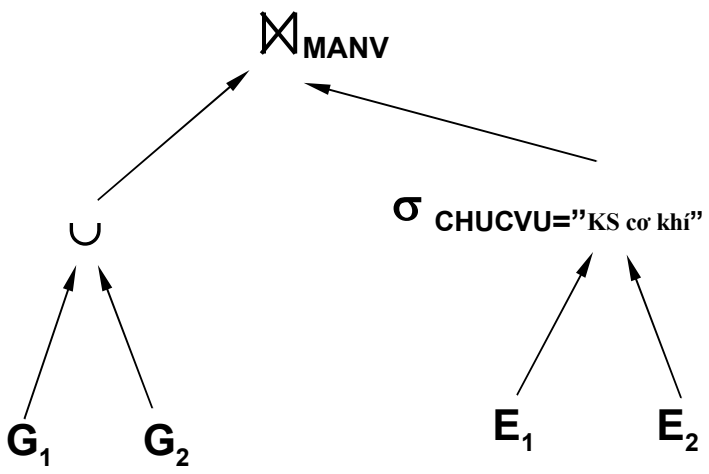
Để rút gọn các truy vấn trên phân mảnh gián tiếp này, phép nối sẽ đưa ra quan hệ rỗng nếu các tân từ phân mảnh mâu thuẫn nhau.

Ví dụ tân từ G_1 và E_2 mâu thuẫn nhau, nên $G_1 \ltimes E_2 = \phi$.

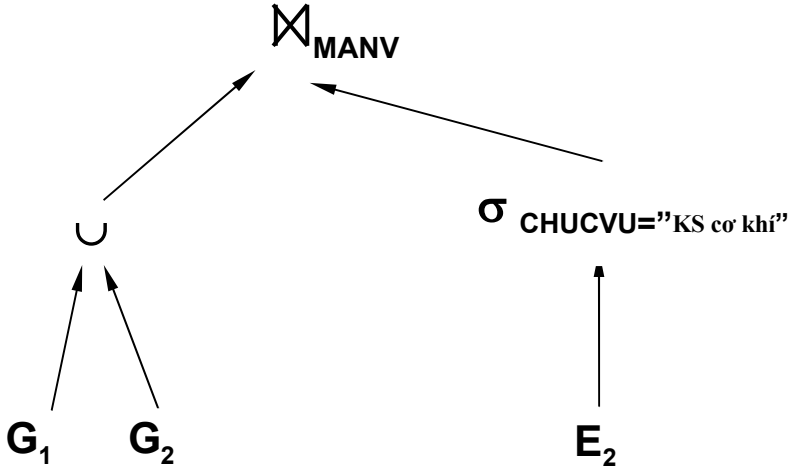
4.3 Xử lý truy vấn trong môi trường phân tán

Ví dụ: Xét truy vấn

```
SELECT      *
FROM        E, G
WHERE       G.MANV=E.MANV
      AND   CHUCVU=”KS cơ khí”
```

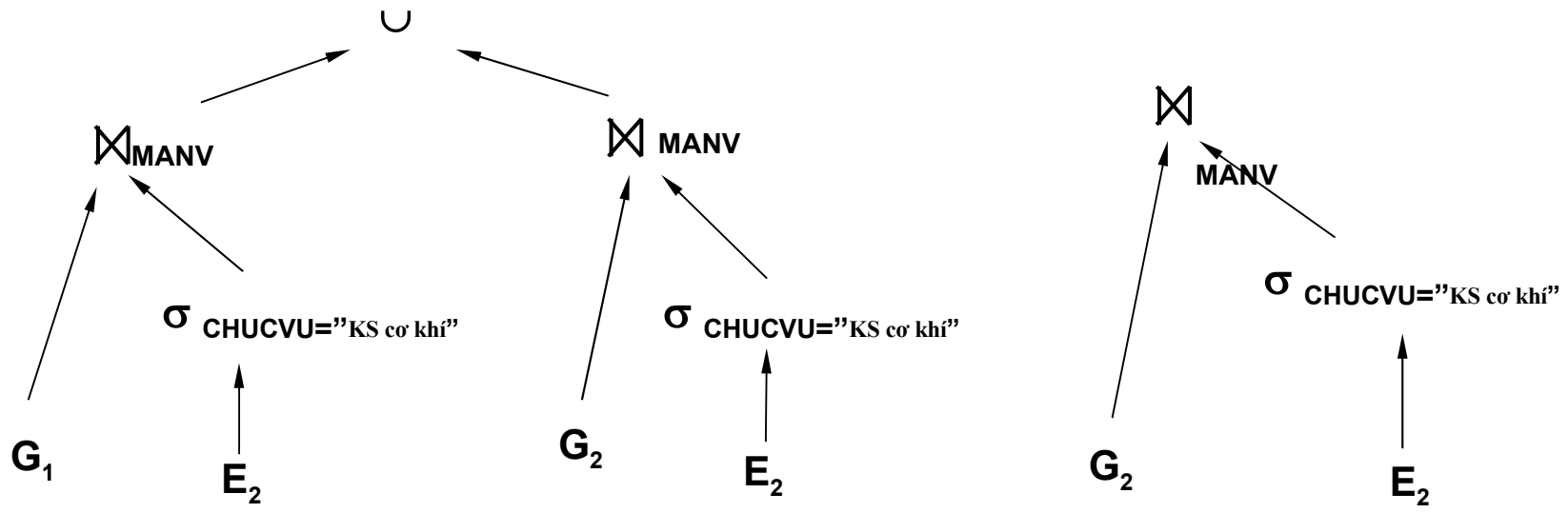


(a) Truy vấn ban đầu



(b) Truy vấn sau khi đẩy phép chọn xuống

4.3 Xử lý truy vấn trong môi trường phân tán



(c) Truy vấn sau khi đẩy phép hợp lên **(d) Truy vấn đã rút gọn**

Hình 4.10: Rút gọn của phân mảnh gián tiếp

Nhận xét:

- Truy vấn ban đầu trên các mảnh E_1, E_2, G_1 và G_2 tương ứng hình 4.10a.
- Bằng cách đẩy phép chọn xuống các mảnh E_1 và E_2, được truy vấn rút gọn ở hình 4.10b.
- Phân phối các phép nối với phép hợp, chúng ta thu được cây hình 4.10c.
- Cây con bên trái đưa ra một quan hệ rỗng, nên cây rút gọn có được trong hình 4.10d

4.2.2.4 Rút gọn theo phân mảnh hỗn hợp

- Sự phân mảnh hỗn hợp là sự kết hợp giữa phân dọc và phân mảnh ngang.
- Mục đích của phân mảnh hỗn hợp là hỗ trợ các truy vấn liên quan đến phép chiếu, phép chọn, phép nối
- Chương trình định vị cho một quan hệ đã phân mảnh hỗn hợp sử dụng phép hợp và phép nối của các mảnh.

Ví dụ: Xét quan hệ E được phân mảnh hỗn hợp như sau:

$E_1=\sigma_{MANV \le \text{"E4"}}(\Pi_{MANV,TENNV}(E))$, $E_2=\sigma_{MANV > \text{"E4"}}(\Pi_{MANV,TENNV}(E))$

$E_3=\Pi_{MANV,CHUCVU}(E)$

Chương trình định vị là: $E = (E_1 \cup E_2) \bowtie_{MANV} E_3$

4.3 Xử lý truy vấn trong môi trường phân tán

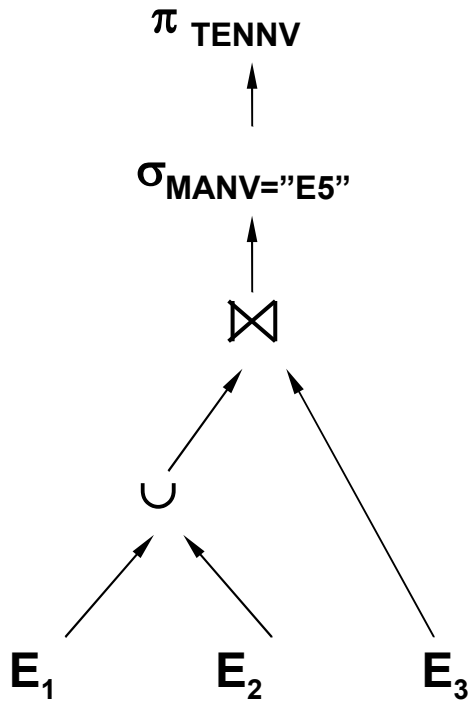
Các truy vấn trên các mảnh hỗn hợp có thể được rút gọn bằng cách kết hợp các luật sử dụng trong phân mảnh ngang nguyên thủy, phân mảnh dọc, phân mảnh ngang gián tiếp, tương ứng như sau:

1. Loại bỏ các quan hệ rỗng sinh bởi sự mâu thuẫn giữa các phép chọn trên các phân mảnh ngang.
2. Loại bỏ các quan hệ vô ích sinh bởi các phép chiếu trên các phân mảnh dọc.
3. Phân phối các phép nối với các phép hợp để tách và loại bỏ các phép nối vô ích.

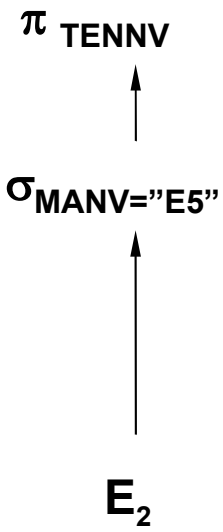
Ví dụ: Truy vấn trong SQL dưới đây minh hoạ việc ứng dụng các luật (1), (2) đến sự phân mảnh dọc_ngang của quan hệ E cho trên thành E_1, E_2 và E_3.

4.3 Xử lý truy vấn trong môi trường phân tán

SELECT TENNV
FROM E
WHERE MANV="E_5"



(a) Truy vấn ban đầu



(b) Truy vấn đã rút gọn

Hình 4.11: Rút gọn của phân mảnh hỗn hợp

4.4 Tối ưu hóa truy vấn trong CSDL phân tán

1. Truy vấn thu được từ giai đoạn phân rã và định vị dữ liệu có thể được thực hiện một cách đơn giản bằng việc thêm vào các thao tác truyền thông.
2. Việc hoán vị thứ tự các phép toán trong một câu truy vấn có thể cung cấp nhiều chiến lược tương đương khác nhau.
3. Bài toán xác định cây truy vấn tối ưu là NP-khó. Thông thường bộ tối ưu tìm tìm một chiến lược gần tối ưu và tránh các chiến lược "tồi".
4. Đầu ra của bộ tối ưu là một lịch trình được tối ưu bao gồm truy vấn đại số được xác định trên các mảnh và các phép toán truyền thông hỗ trợ việc thực hiện truy vấn trên các trạm.
5. Để chọn lựa được một chiến lược tối ưu nói chung, bộ tối ưu phải xác định chi phí thực hiện câu truy vấn.
6. Chi phí thực hiện là tổ hợp có trọng số của chi phí truyền thông, chi phí I/O và chi phí CPU.

4.4.1 Mô hình chi phí của bộ tối ưu hóa truy vấn

Chi phí của một chiến lược thực hiện phân tán có thể được biểu diễn hoặc theo tổng chi phí hoặc theo thời gian trả lời.

- **Tổng chi phí** là tổng của tất cả các thành phần chi phí. bao gồm chi phí truyền thông, chi phí I/O và chi phí CPU.

Tuy nhiên, để đơn giản ta bỏ qua chi phí xử lý địa phương (I/O, CPU), coi chi phí truyền thông là trọng yếu.

- **Thời gian trả lời truy vấn** là thời gian được tính từ khi bắt đầu xử lý đến khi hoàn thành truy vấn.

2. Công thức chung cho sự xác định tổng chi phí:

Tæng chi phÝ: tæng cña tÊt c¶ c,c chi phÝ C_{CPU}, $C_{I/O}$. C_{MSG}

Total_cost= C_{CPU} * #instr + $C_{I/O}$ * #I/O_S + C_{MSG} * #msgs + + C_{TR} * #bytes

Trong đó: **Total_cost**: tổng chi phí

C_{CPU}: chi phí của một lệnh CPU

$C_{I/O}$: chi phí của một xuất/nhập đĩa

C_{MSG}: chi phí của việc khởi đầu và nhận một thông báo.

C_{TR}: chi phí truyền một đơn vị dữ liệu từ trạm này đến tram khác, ta xem **C_{TR}** như là một hằng số.

#instr: tổng tất cả các lệnh CPU ở các trạm

#I/O_S: số lần xuất/nhập đĩa

Trong công thức:

Total_cost= C_{CPU}* #instr + $C_{I/O}$*#I/OS + C_{MSG} *#msgs +

+ C_{TR} *#bytes

- Hai thành phần chi phí đầu (**C_{CPU},$C_{I/O}$**) là *chi phí địa phương*.
- Hai thành phần chi phí sau (**C_{MSG}, C_{TR}**) là *chi phí truyền thông*.
- Chi phí truyền thông để chuyển #byte dữ liệu từ trạm này đến trạm khác được giả thiết là một hàm tuyến tính theo số #bytes được truyền đi, được xác định bởi công thức

CC(#byte)= C_{MSG} + C_{TR} * bytes

1. Công thức chung cho sự xác định thời gian trả lời

$$Response\_time = C_{CPU} * seq\_\#instr + C_{I/O} * seq\_\#I/O_S + C_{MSG} * seq\_\#msgs + C_{TR} * seq\_\#bytes$$

Trong ®ã:

seq_#x (x cã thÓ lµ sè lÖnh cña CPU, I/O, sè th«ng b¸o, sè byte) lµ sè lín nhÊt cña x khi thùc hiÖn truy vÊn mét c¸ch tuÇn tù.

Trong đó: **Response_time**: thời gian trả lời truy vấn

C_{CPU}: chi phí của một lệnh CPU

$C_{I/O}$: chi phí của một xuất/nhập đĩa

C_{MSG}: chi phí của việc khởi đầu và nhận một thông báo.

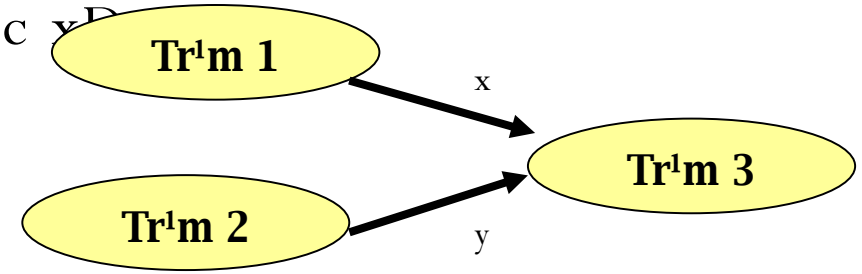
C_{TR}: chi phí truyền một đơn vị dữ liệu từ trạm này đến trạm khác

#instr: tổng tất cả các lệnh CPU ở các trạm

#I/OS: số lần xuất/nhập đĩa

#msgs: số thông báo

VÝ dô: Minh ho¹ sù kh¸c nhau gi÷a ***tæng chi phÝ*** vµ ***thêi gian tr¶ lêi***, trong ®ã m¸y tÝnh tr¶ lêi truy vÊn t¹i tr¹m 3 víi d÷ liÖu tõ tr¹m 1 vµ 2, ë ®©y chØ cã chi phÝ truyÒn th«ng ®-îc x



H×nh 4.12: VÝ dô cña sù biÕn ®æi 1 truy vÊn

Gi¶ sö, C_{MSG} vµ C_{TR} ®-îc biÓu thÞ theo ®¬n vÞ thêi gian. Tæng chi phÝ truyÒn x ®¬n vÞ tõ tr¹m 1 ®Õn tr¹m 3 vµ y ®¬n vÞ tõ tr¹m 2 ®Õn tr¹m 3 lµ:

$$\mathbf{Total\_cost = C_{MSG} + C_{TR}*x + C_{MSG} + C_{TR}*y = 2C_{MSG} + C_{TR}*(x+y)}$$

V× viÖc truyÒn d÷ liÖu cã thÓ ®-îc thùc hiÖn song song nªn thêi gian tr¶ lêi cña truy vÊn lµ

$$\mathbf{Response\_time = max\{C_{MSG} + C_{TR}*x, C_{MSG} + C_{TR}*y\}}$$

4.4.2 Các thống kê dữ liệu

•Yếu tố chính ảnh hưởng đến hiệu suất của một chiến lược thực thi là kích thước của các quan hệ trung gian sinh ra trong quá trình thực hiện.

•Khi phép toán tiếp theo đặt tại một trạm khác, quan hệ trung gian phải được truyền trên mạng.

•Do đó để tối thiểu hoá khối lượng dữ liệu truyền đi, điều quan tâm đầu tiên là đánh giá kích thước kết quả trung gian của các phép toán đại số quan hệ.

•Đánh giá này dựa trên các thông tin thống kê về các quan hệ cơ sở và các công thức ước tính lực lượng của kết quả các phép toán quan hệ.

Mục đích của thống kê dữ liệu:

- Xác định kích thước của các quan hệ trung gian sinh ra trong quá trình thực hiện câu truy vấn
- Xác định chi phí truyền thông cho các đại lượng trung gian

Một số ký hiệu

Cho quan hệ R xác định trên tập thuộc tính $A=\{A_1, ..., A_n\}$. R được phân mảnh thành $R_1, R_2, ..., R_r$.

- **length(A_i)**: độ dài (byte) của thuộc tính A_i ,$A_i \in R$,
- **card($\pi_{Ai}(R_j)$)**: lực lượng của phép chiếu của mảnh R_j lên thuộc tính A_i (số giá trị phân biệt trên thuộc tính A_i) .
- **max(A_i):** giá trị cực đại của thuộc tính A_i trong Dom(A_i)
- **min(A_i):** giá trị cực tiểu của thuộc tính A_i trong Dom(A_i)
- **card(dom(A_i)):** lực lượng của thuộc tính A_i
- **card(R_i)):** số các bộ trong mảnh R_i

Ngoài ra, dữ liệu thống kê cũng bao gồm *hệ số chọn của phép nối* (**SF_J**) đối với một số cặp đại số quan hệ, hệ số **SF_J** của quan hệ R và S là một số thực giữa 0 và 1, được xác định bởi:

$$SF_J = \frac{card(R \bowtie S)}{card(R)*card(S)}$$

- Hệ số **SF_J** nhỏ thì phép nối có tính chọn tốt, ngược lại có tính chọn tồi.
- Các thống kê này có lợi để đánh giá kích thước của quan hệ trung gian.
- Kích thước một quan hệ trung gian R được xác định bởi **size(R) = card(R)*length(R).** Trong đó,
  - + **length(R)** là độ dài (số byte) của mỗi bộ trong R, được tính theo độ dài các thuộc tính của nó,
  - + **card(R)** là số các bộ của R được tính theo công thức ở phần tiếp theo.

4.4.3 Lực lượng của các kết quả trung gian

Phần này sẽ đưa ra các công thức để ước tính lực lượng kết quả các phép toán cơ sở của đại số quan hệ (phép chọn, phép chiếu, phép tích Decartes, nối, nửa nối, phép hợp và phép trừ). Các toán hạng quan hệ được ký hiệu bởi R và S. Hệ số chọn của một phép toán SF_{OP}, (OP biểu thị phép toán) là tỷ lệ giữa các bộ của một toán hạng quan hệ tham gia vào kết quả của phép toán.

Ví dụ:

- SF_J : hệ số chọn của phép nối
- SF_S : hệ số chọn của phép chọn

1. Phép chọn

$card(\sigma(R)) = SF_S(F) * card(R)$

Trong đó $SF_S(F)$ phụ thuộc vào công thức chọn và có thể tính như sau, với $p(A_i)$, $p(A_j)$ là các tân từ tương ứng với các thuộc tính A_i, A_j.

$$SF_S(A=value) = \frac{1}{Card(\pi_A(R))}$$

$$SF_S(A>value) = \frac{max(A) - value}{max(A)-min(A)}$$

$$SF_S(A>value) = \frac{Value - min(A)}{max(A)-min(A)}$$

$SF_S(p(A_i) \wedge p(A_j)) = SF_S(p(A_i)) * SF_S(p(A_j))$

$SF_S(p(A_i) \vee p(A_j)) = SF_S(p(A_i)) + SF_S(p(A_j)) - SF_S(p(A_i)) * SF_S(p(A_j))$

$SF_S(A \in \{value\}) = SF_S(A=value) * card(\{value\})$

2. Phép chiếu

Phép chiếu có thể có hoặc không loại bỏ các bản sao, ở đây chỉ xét phép chiếu loại bỏ các bản sao.

Lực lượng quan hệ kết quả của một phép chiếu tùy ý là khó đánh giá chính xác, vì tương quan giữa thuộc tính chiếu là thường không biết. Tuy nhiên, có hai trường hợp tầm thường nhưng đặc biệt có lợi:

- Nếu phép chiếu của R trên một thuộc tính đơn A thì lực lượng được tính đơn giản là số các bộ khi phép chiếu được thực hiện.
- Nếu một trong các thuộc tính chiếu là khoá của R, thì card($\pi_A(R)$) = card(R) và card($R \times S$) = card(R) * card(S)

3. Phép nối

•Không có một cách tổng quát để xác định lực lượng của một phép nối nếu không có các thông tin thêm.

•Cận trên của lực lượng của phép nối chính là lực lượng của tích Decartes.

•Tuy nhiên, có một số trường hợp xuất hiện thường xuyên và việc đánh giá là đơn giản:

- Nếu $R \bowtie_{AB} S$ với $A \in R$, $B \in S$, trong đó A là khoá của R, B là khoá ngoài của S, thì lực lượng của kết quả xấp xỉ là: $card(R \bowtie_{AB} S) = card(R)$

•Với các phép nối khác, lực lượng của kết quả là:

$$card(R \bowtie S) = SF_J * card(R) * card(S)$$

4. Phép nửa nối

Hệ số chọn của phép nửa nối (SF_{SJ}) xấp xỉ là:

$$SF_{SJ}(R \ltimes S) = \frac{Card(\pi_A(S))}{Card(dom[A])}$$

Công thức này chỉ phụ thuộc vào thuộc tính A của S, nên thường được gọi là hệ số chọn thuộc tính A của S, ký hiệu $SF_{SJ}(S.A)$ và là hệ số chọn của S.A trên bất cứ thuộc tính nối khác. Vì thế, lực lượng của phép nối được tính như sau:

$$card(R \ltimes AS) = SF_{SJ}(S.A) * card(R)$$

5. Phép hợp

•Rất khó đánh giá số lượng của R∪S, vì các bộ giống nhau giữa R và S bị loại bỏ bởi phép hợp.

•Ở đây chúng ta chỉ đưa ra công thức tính:

-cận trên của card(R∪S) bằng card(R)+card(S),

-cận dưới của card(R∪S) bằng max{card(R),card(S)}

(giả sử R và S không chứa các bộ lặp).

6. Phép trừ

Cũng như phép hợp ở đây chỉ đưa ra cận trên và cận dưới, cận trên của card(R-S) là card(R), cận dưới là 0.

4.4 Tối ưu hóa trong CSDL phân tán

Ví dụ Xét hai quan hệ trong cơ sở dữ liệu của công ty máy tính:

E=NHANVIEN (MANV, TENNV, CHUCVU) và

G=HOSO (MANV, MADA, NHIEMVU, THOIGIAN).

Với câu truy vấn "*Cho biết tên các nhân viên hiện đang quản lý một dự án*". Ta có câu truy vấn SQL tương ứng là:

```
SELECT    TENNV
FROM      E, G
WHERE     E.MANV=G.MANV
          AND  NHIEMVU="Quản lý"
```

Hai truy vấn đại số tương đương với truy vấn trên là:

$\Pi_{TENNV}(\sigma_{NHIEMVU="Quản\ lý" \wedge E.MANV=G.MANV}(E \times G))$ (1)

và $\Pi_{TENNV}(E \bowtie_{MANV}(\sigma_{NHIEMVU="Quản\ lý"} G))$ (2)

Rõ ràng truy vấn (2) tránh được khỏi phải tích số của E và G, nên dùng ít phép tính tài nguyên hơn truy vấn (1)

Câu hỏi cuối chương

1. Mục đích của tối ưu hoá truy vấn trong CSDL phân tán
2. Chức năng của tối ưu hoá truy vấn phân tán
3. Các phương pháp xử lý truy vấn cơ bản
4. Ý tưởng của thuật toán Ingres. Ví dụ
5. Sơ đồ phân lớp chung cho cho xử lý truy vấn phân tán
6. Định vị dữ liệu phân tán-Tối ưu hoá cục bộ

HẾT CHƯƠNG 4