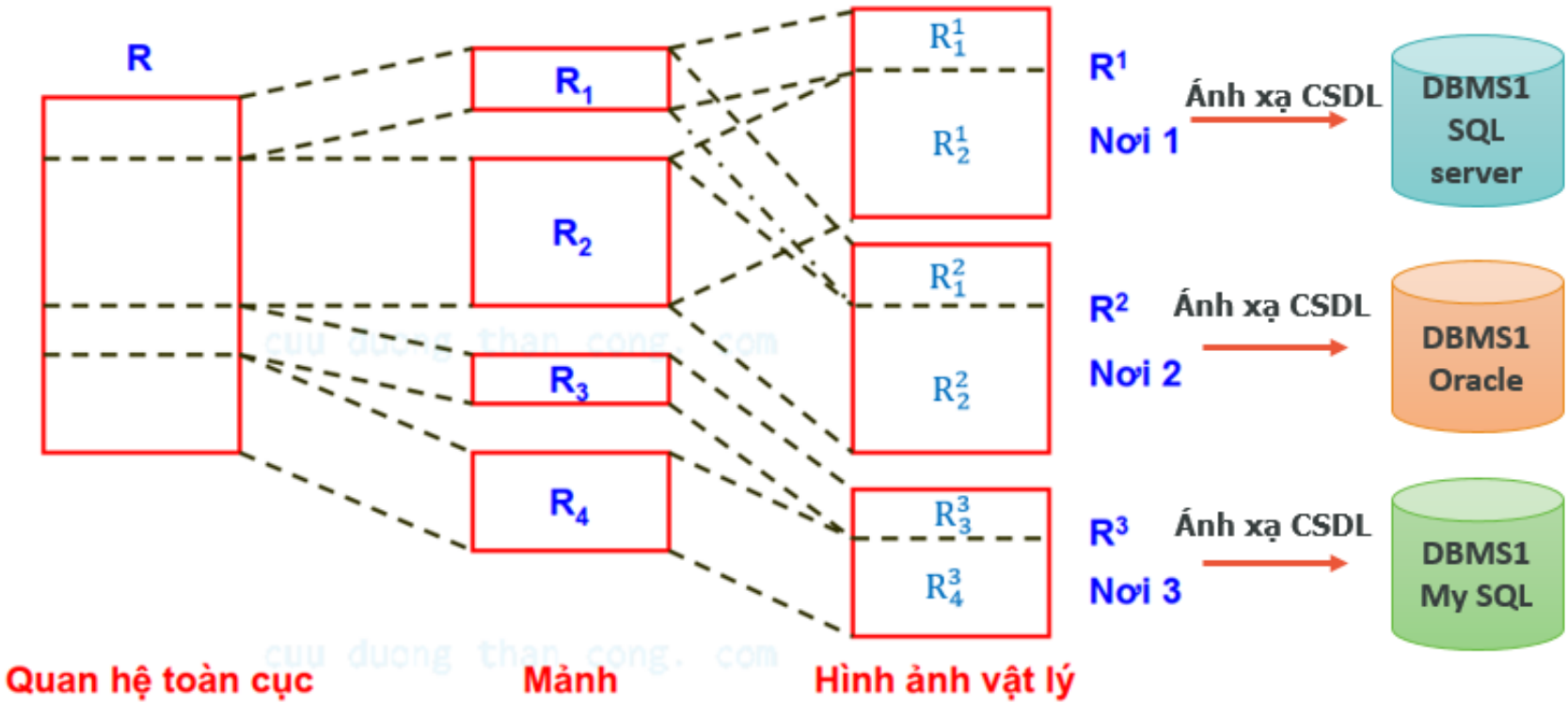
CHƯƠNG 3
Các mức trong suốt phân tán

Nội dung

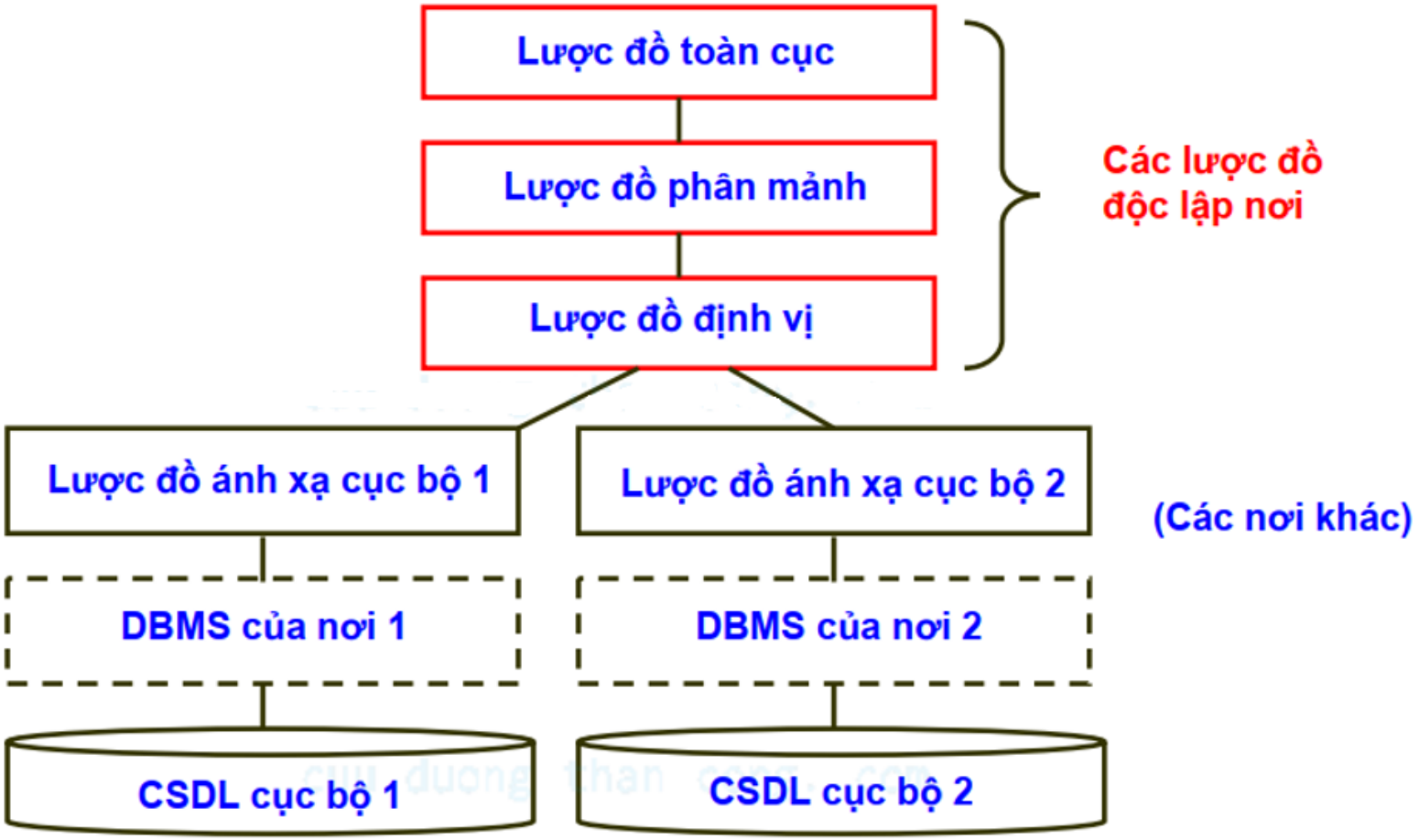
- Kiến trúc tham khảo của CSDL phân tán.
- Phân mảnh dữ liệu.
- Các điều kiện đúng đắn để phân mảnh dữ liệu.
- Phân mảnh ngang chính.
- Phân mảnh ngang dẫn xuất.
- Phân mảnh dọc.
- Phân mảnh hỗn hợp.
- Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc.
- Tính trong suốt phân tán dùng cho ứng dụng cập nhật.
- Các tác vụ cơ bản truy xuất CSDL phân tán.

Kiến trúc tham khảo của CSDL phân tán

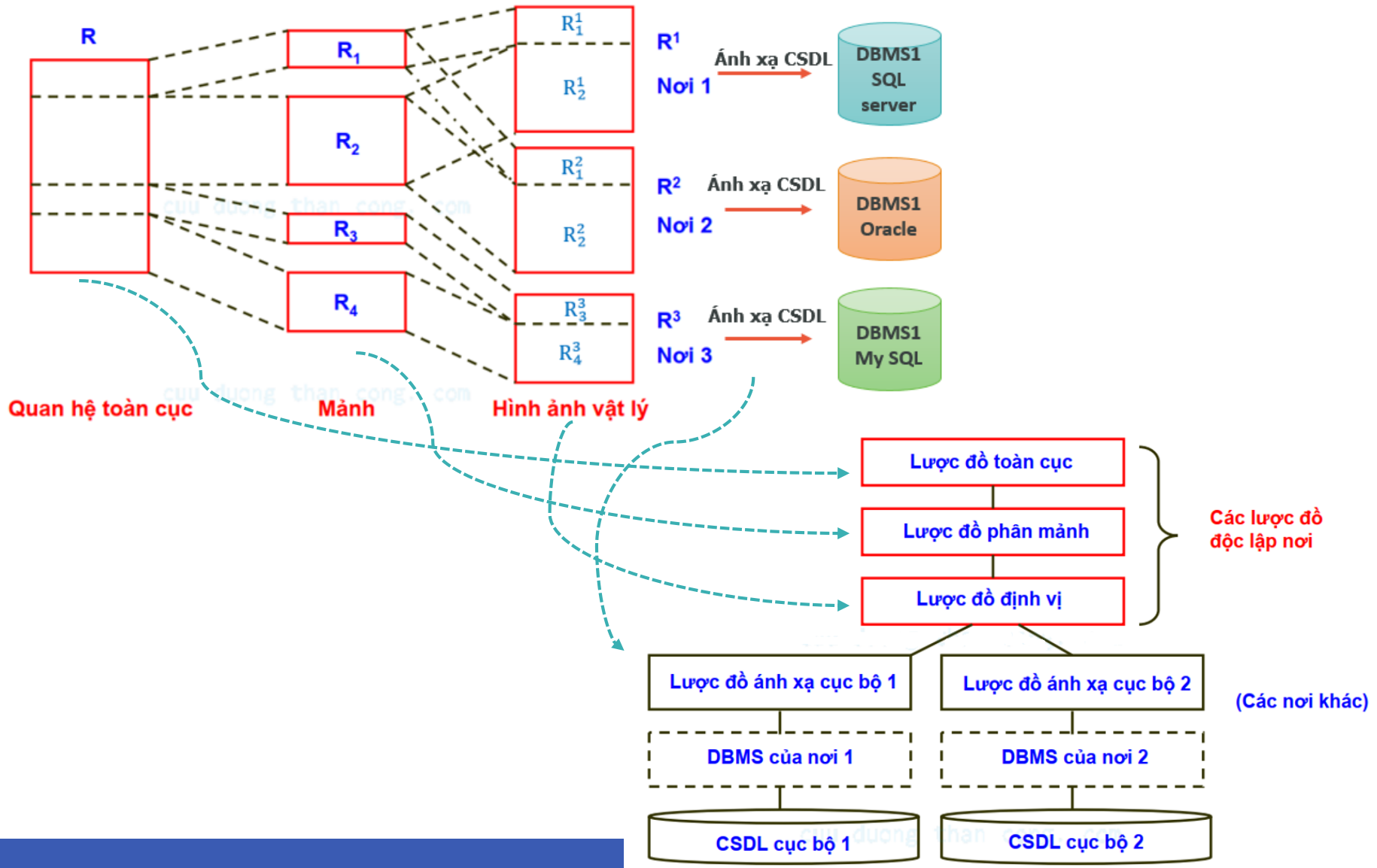


Các mảnh và các hình ảnh vật lý của một quan hệ toàn cục

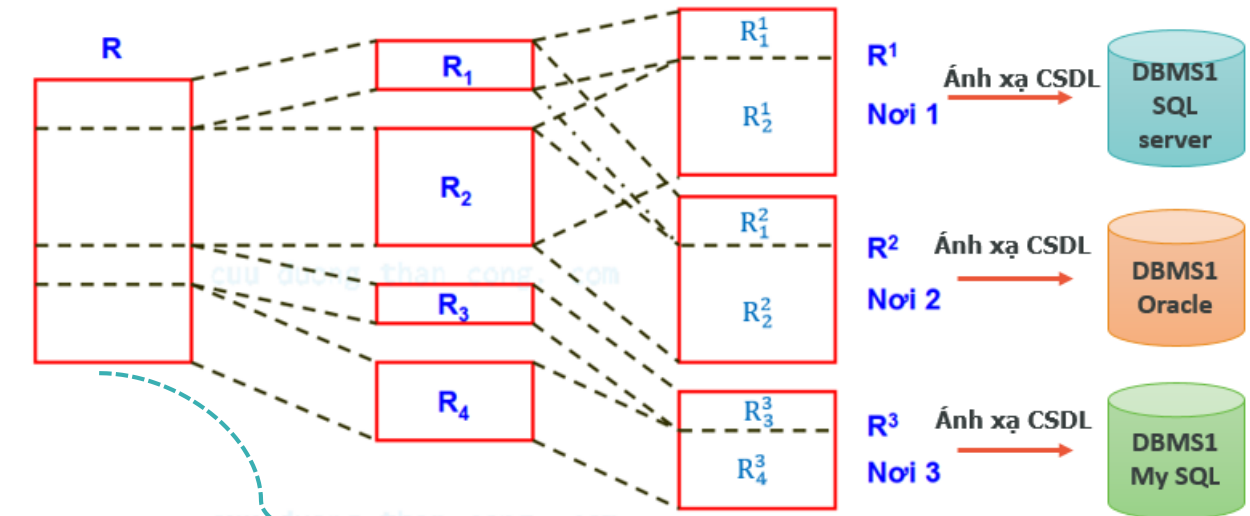
Kiến trúc tham khảo của CSDL phân tán



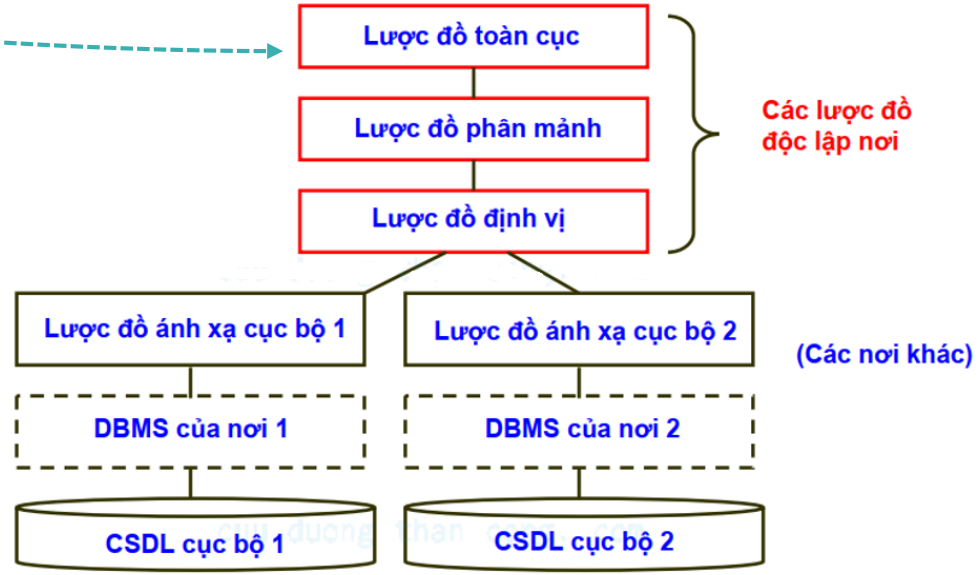
Kiến trúc tham khảo của CSDL phân tán



Kiến trúc tham khảo của CSDL phân tán



Lược đồ toàn cục (global schema)

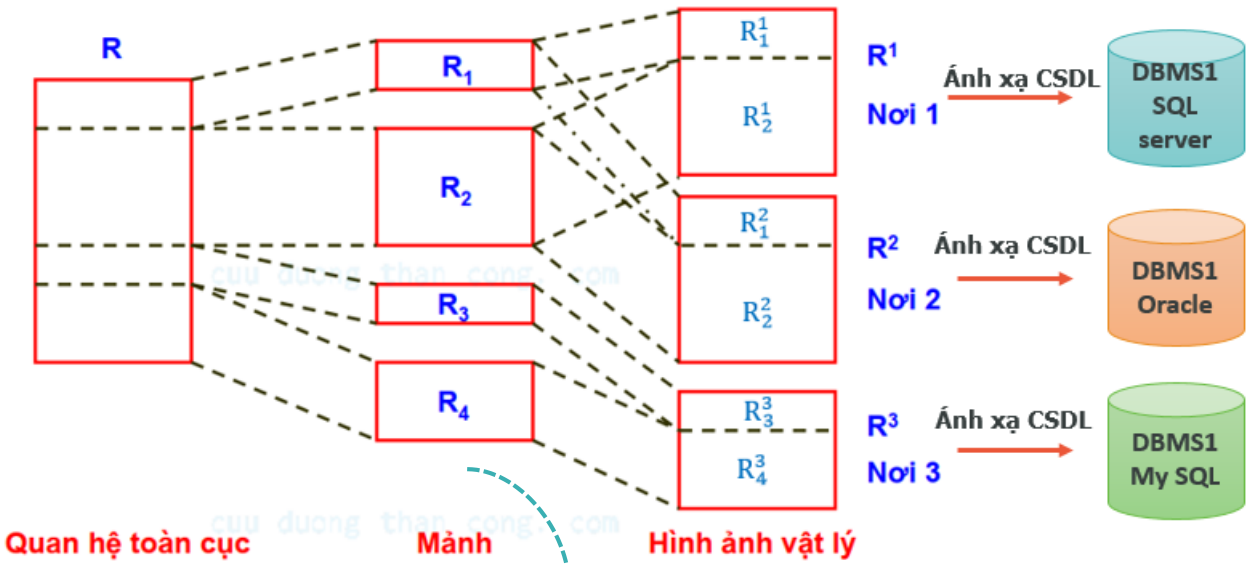


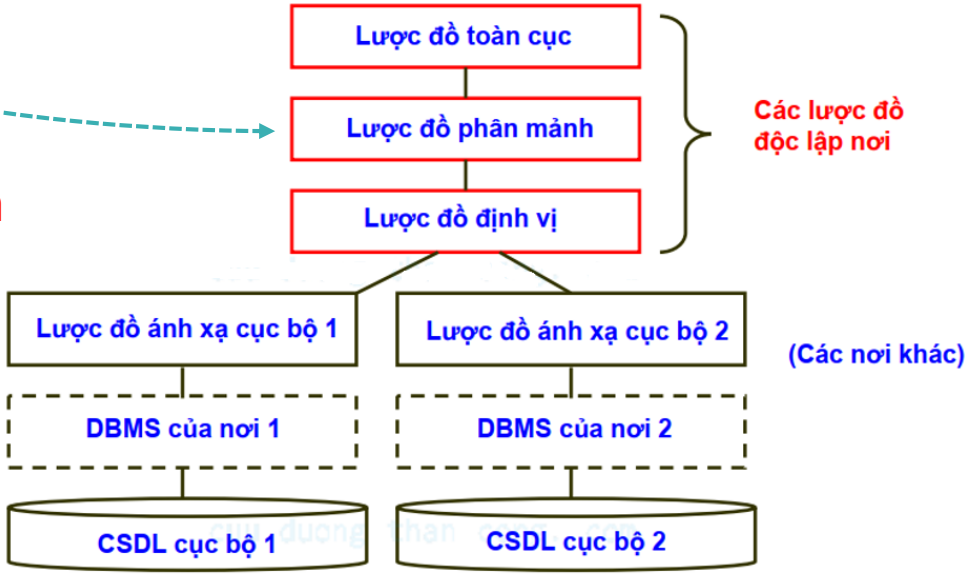
Kiến trúc tham khảo của CSDL phân tán

❖ Lược đồ toàn cục (global schema)

- Định nghĩa tất cả các dữ liệu chứa trong CSDL PT cũng như các dữ liệu không được phân tán ở các trạm trong hệ thống
- Được định nghĩa hoàn toán giống như CSDL tập trung
- Trong mô hình quan hệ thì lược đồ toàn cục bao gồm định nghĩa 1 tập hợp các quan hệ toàn cục (global relation)

Kiến trúc tham khảo của CSDL phân tán





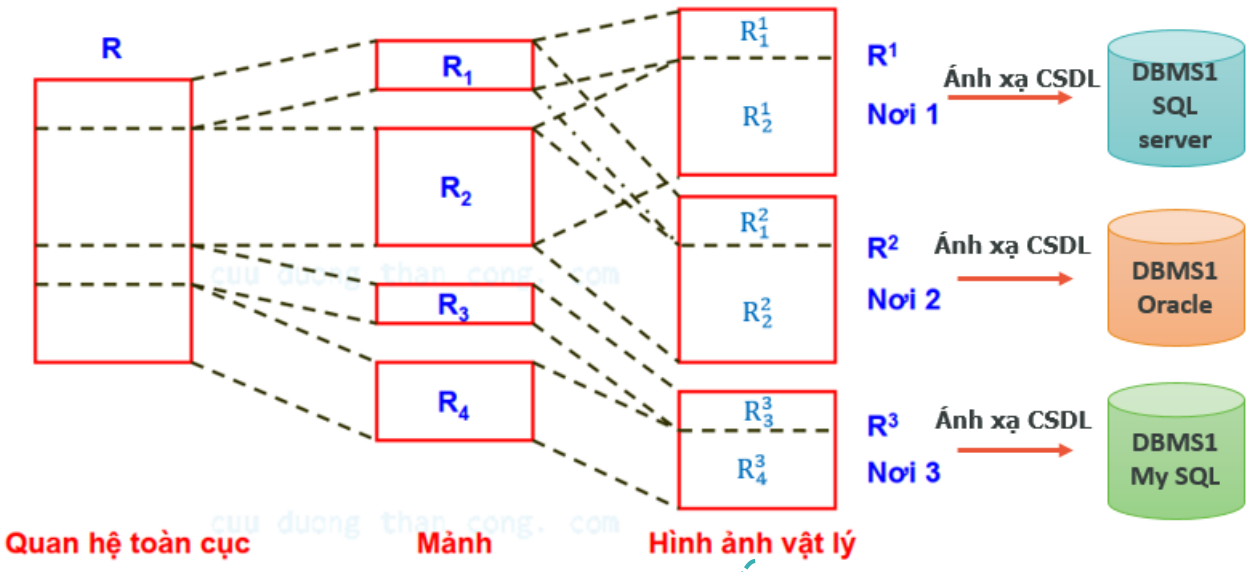
Lược đồ phân mảnh (fragmentation schema)

Kiến trúc tham khảo của CSDL phân tán

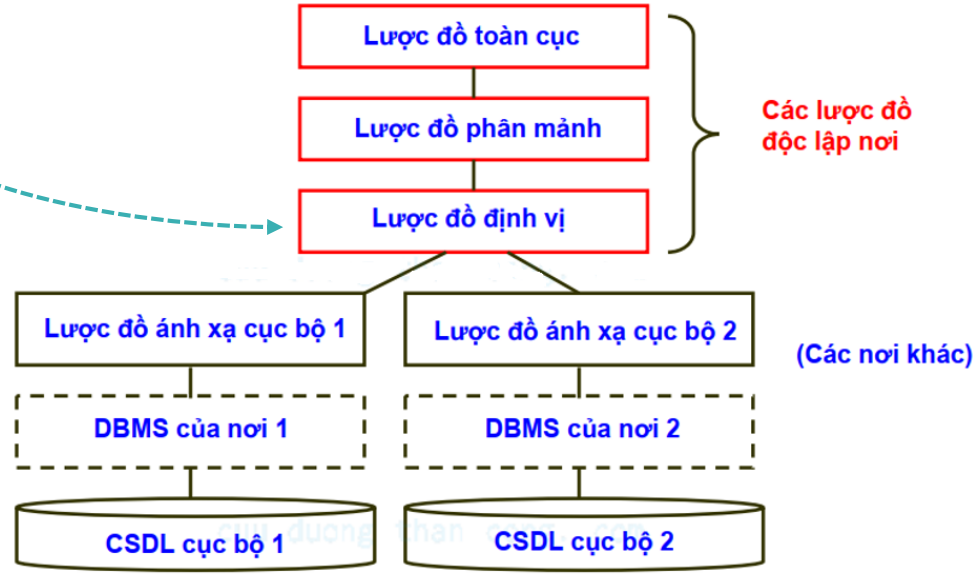
❖ Lược đồ phân mảnh (fragmentation schema)

- Mỗi quan hệ toàn cục có thể được phân thành nhiều phần không phủ lấp nhau → được gọi là **mảnh** (**fragment**).
- Ánh xạ giữa các quan hệ toàn cục và các mảnh được gọi là lược đồ phân mảnh. (ánh xạ 1-nhiều)
- Các mảnh được mô tả bằng tên của quan hệ toàn cục cùng với chỉ mục. ***Ví dụ: R_i được hiểu là mảnh thứ i của quan hệ R***
- Có nhiều cách khác nhau để thực hiện việc phân chia fragments

Kiến trúc tham khảo của CSDL phân tán



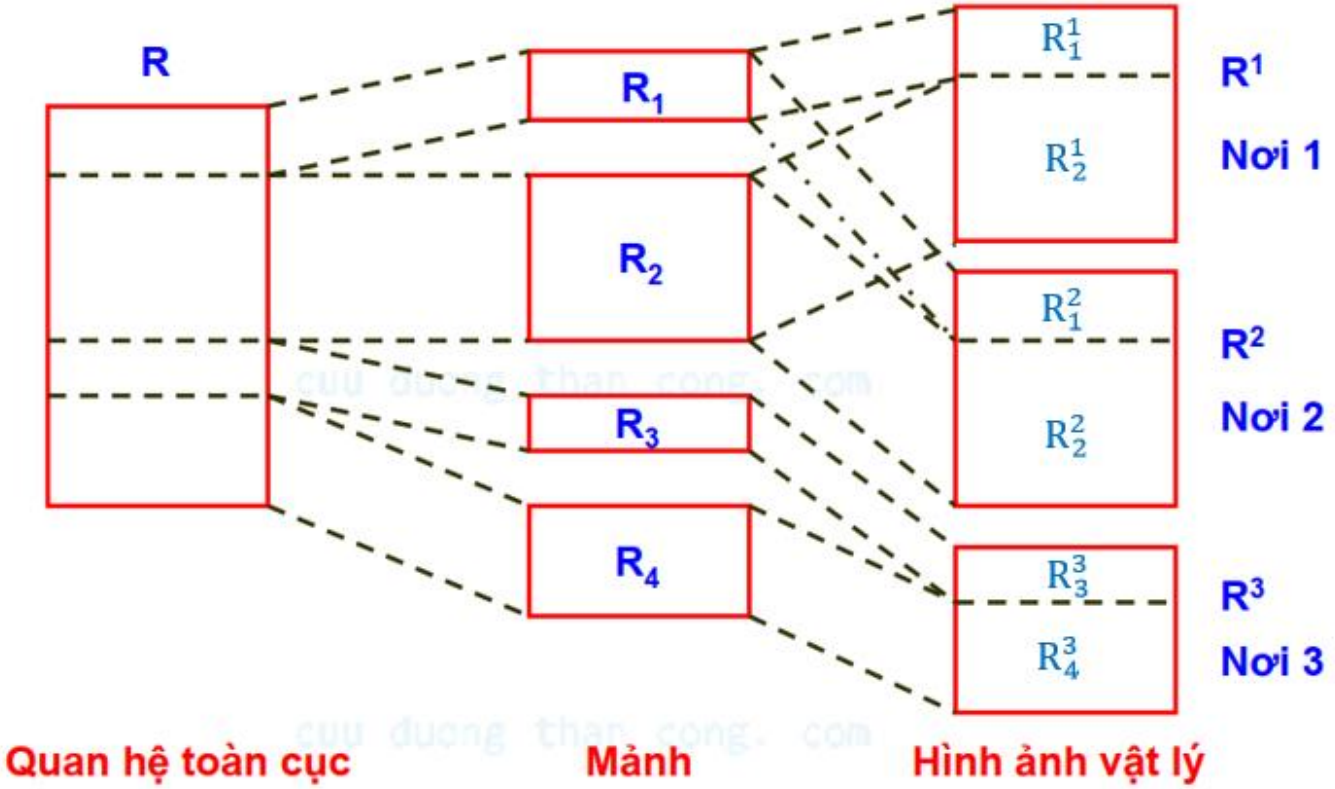
Lược đồ định vị (allocation schema)



❖ Lược đồ định vị (allocation schema)

- Xác định một mảnh được đặt tại nơi nào (được định vị vật lý trên một hay nhiều trạm)
- **Hình ảnh vật lý (physical image):** Các mảnh thuộc quan hệ toàn cục R được đặt tại cùng 1 nơi j tạo thành 1 hình ảnh vật lý của quan hệ toàn cục R tại j, ký hiệu $\mathbf{R^j}$
- **Bản nhân (replica):** một mảnh được đặt tại một nơi cho trước, ký hiệu bằng tên quan hệ toàn cục và hai chỉ số (chỉ số mảnh, chỉ số nơi. Ví dụ R_4^3 là bản nhân của mảnh R_4 đặt tại nơi 3

Kiến trúc tham khảo của CSDL phân tán



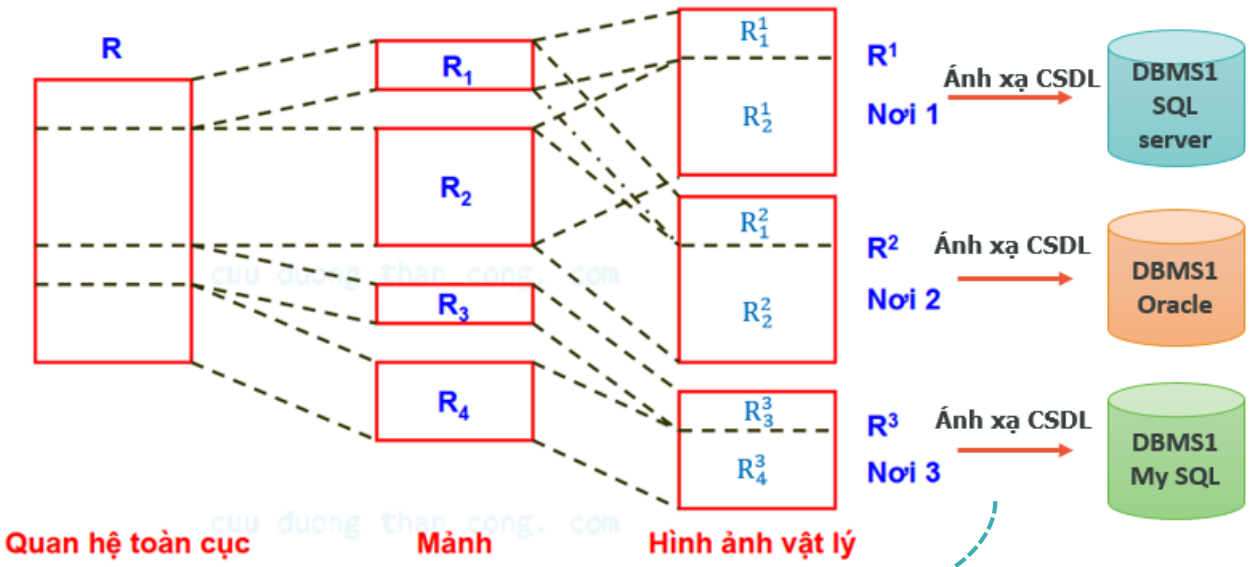
Các mảnh và các hình ảnh vật lý của một quan hệ toàn cục

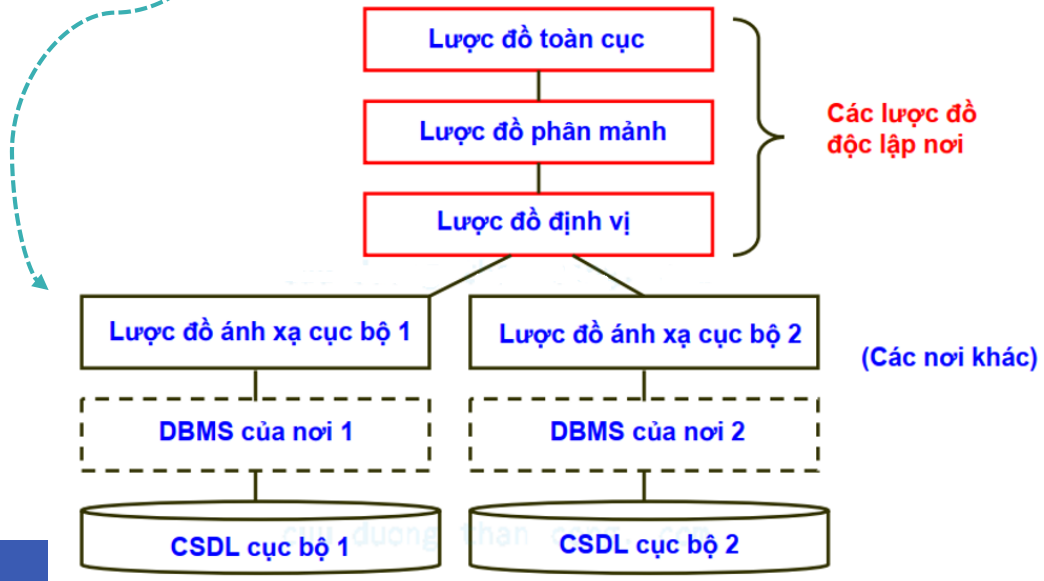
Quan hệ toàn cục R được phân thành 4 mảnh R_1, R_2, R_3, R_4, 4 mảnh này được đặt dư thừa tại 3 nơi tạo thành 3 hình ảnh vật lý R^1, R^2, R^3

Kiến trúc tham khảo của CSDL phân tán

- Ba mức trên (toàn cục, phân mảnh, định vị) là độc lập vị trí (site independent) → không phụ thuộc vào mô hình DBMS cục bộ.
- Sự tách biệt giữa phân mảnh dữ liệu với khái niệm định vị dữ liệu
- Biết được dữ liệu dư thừa

Kiến trúc tham khảo của CSDL phân tán





Lược đồ ánh xạ cục bộ (local mapping schema)

Kiến trúc tham khảo của CSDL phân tán

- **Lược đồ ánh xạ cục bộ (local mapping schema):** ở các nơi (cục bộ) dùng để ánh xạ các hình ảnh vật lý vào các đối tượng được thao tác bởi các DBMS cục bộ

Các mức trong suốt trong CSDLPT

- Trong suốt phân mảnh (fragmentation transparency)
- Trong suốt vị trí (location transparency)
- Trong suốt nhân bản (replication transparency)
- Trong suốt ánh xạ cục bộ (local mapping transparency)
- Không trong suốt

Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt phân mảnh (fragmentation transparency): là mức trong suốt cao nhất cho phép làm việc trên các quan hệ toàn cục**
  - Lược đồ toàn cục
  - Quan hệ toàn cục
  - Cơ sở dữ liệu phân bố trong suốt hoàn toàn. Làm việc trên CSDLPT hoàn toàn giống như làm việc với cơ sở dữ liệu tập trung.

Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt phân mảnh (fragmentation transparency)**

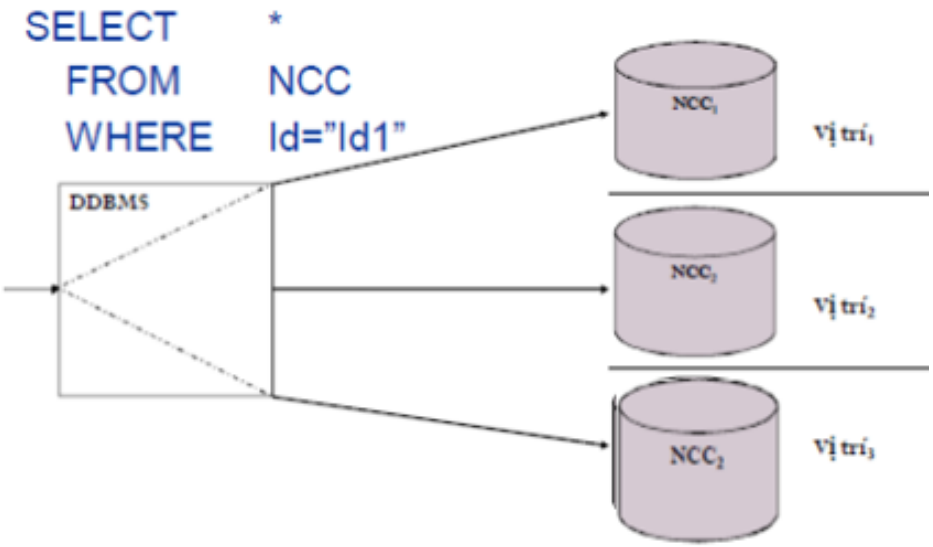
Ví dụ: Xét quan hệ tổng thể NCC (Id, Tên, Tuổi) và các phân mảnh được tách ra từ nó (2 mảnh):

NCC1 (Id, Tên, Tuổi)

NCC2 (Id, Tên, Tuổi)

❖ Trong suốt phân mảnh (fragmentation transparency)

- Giả sử DDBMS cung cấp tính trong suốt về phân mảnh, khi đó ta có thể thấy tính trong suốt này được thể hiện như sau:
- Khi muốn tìm một người có Id="Id1" thì chỉ cần tìm trên quan hệ tổng thể NCC mà không cần biết quan hệ NCC có phân tán hay không.



Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt vị trí (location transparency): làm việc trên các mảnh thay vì các quan hệ toàn cục (tuy nhiên không quan tâm mảnh đặt tại nơi nào)**
  - Lược đồ phân mảnh
  - Mảnh

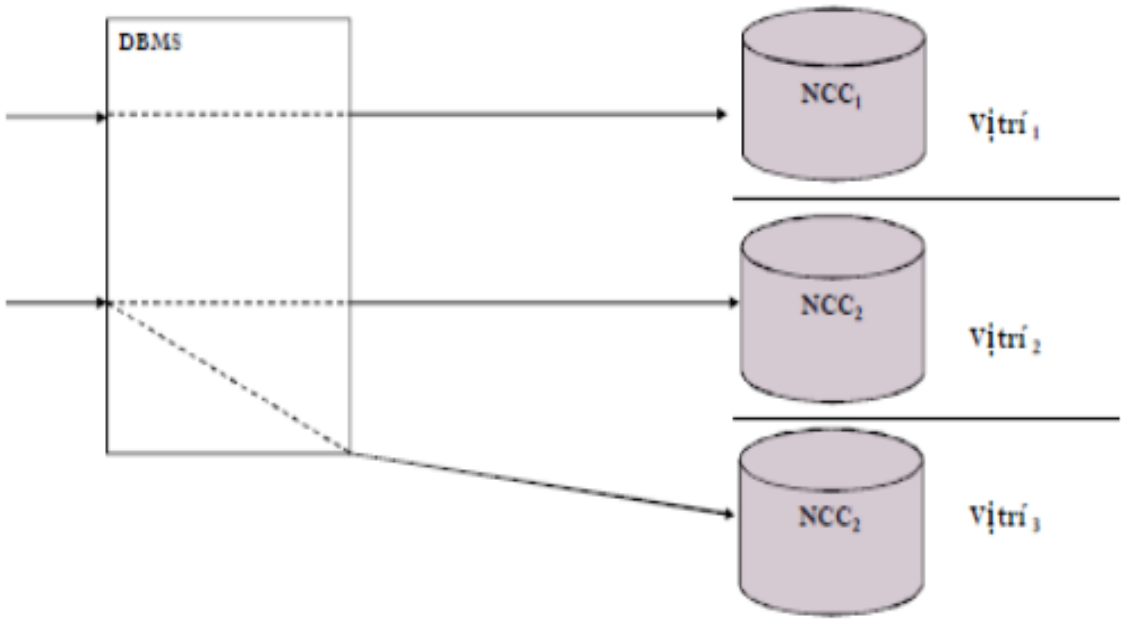
Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt vị trí (location transparency)**

***Ví dụ*: Với quan hệ tổng thể R và các phân mảnh như đã nói ở trên nhưng giả sử rằng DDBMS cung cấp trong suốt về vị trí nhưng không cung cấp trong suốt về phân mảnh**

Xét câu truy vấn *tìm người có* Id=”Id1”

```
SELECT *
FROM NCC1
WHERE Id=”Id1”
IF NOT #FOUND THEN
SELECT *
FROM NCC2
WHERE Id=”Id1”
```



❖ Trong suốt vị trí (location transparency)

- Đầu tiên hệ thống sẽ thực hiện tìm kiếm ở phân mảnh NCC1 và nếu DBMS trả về biến điều khiển #NOT FOUND thì một câu lệnh truy vấn tương tự được thực hiện trên phân mảnh NCC2 ,...
- Ở đây quan hệ NCC2 được sao làm hai bản trên hai vị trí 2 và vị trí 3, ta chỉ cần tìm thông tin trên quan hệ NCC2 mà không cần quan tâm nó ở vị trí nào.

Kiến trúc tham khảo của CSDL phân tán

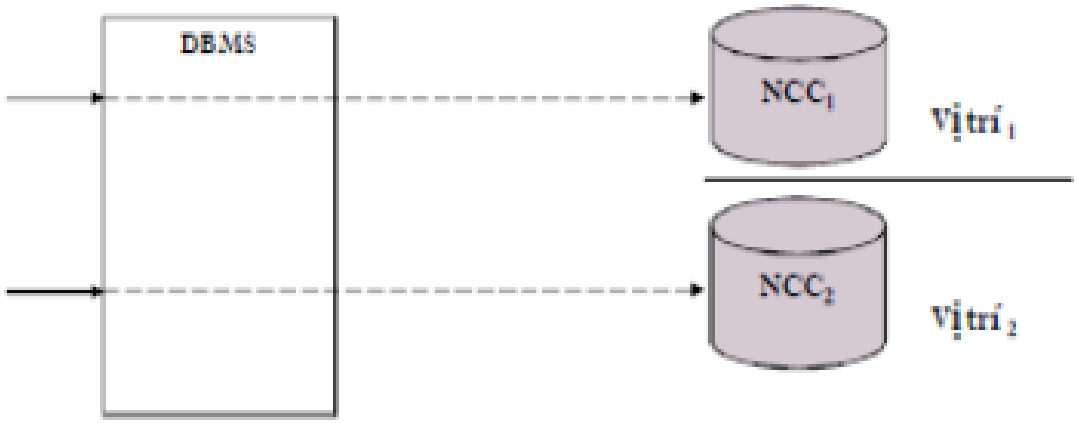
- **Trong suốt nhân bản (replication transparency):**
  - Liên quan chặt chẽ với trong suốt vị trí, người sử dụng không biết có sự nhân bản giữa các mảnh)
  - Bản nhân (replica)
  - Độc lập với các DBMS địa phương

Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt ánh xạ cục bộ (local mapping transparency):**

  sự độc lập giữa các DBMS cục bộ, cho phép không phụ thuộc các mô hình dữ liệu cụ thể của các DBMS cục bộ

  - Lược đồ định vị (allocation schema)
  - Các quan hệ cục bộ (local relation)



Kiến trúc tham khảo của CSDL phân tán

- **Trong suốt ánh xạ cục bộ (local mapping transparency)**
  - Các ứng dụng phải xác định truy nhập vào phân mảnh nào và phân mảnh đó được đặt tại vị trí nào trong hệ cơ sở dữ liệu phân tán.
  - Nhìn thấy quan hệ cục bộ, ko nhìn thấy CSDL vật lý.
  - Ứng dụng tham chiếu đến các đối tượng có các tên độc lập từ các hệ thống cục bộ địa phương.
  - Ứng dụng được cài đặt trên một hệ thống không đồng nhất nhưng được sử dụng như một hệ thống đồng nhất.

❖ Không trong suốt (*No transparency)*

- Lược đồ ánh xạ cục bộ *(local mapping schema)*

Các loại phân mảnh dữ liệu

- **Phân mảnh ngang (horizontal fragmentation)**
  - **Phân mảnh ngang chính (primary horizontal fragmentation)**
  - **Phân mảnh ngang dẫn xuất(derived horizontal fragmentation)**
- **Phân mảnh dọc (vertical fragmentation)**
- **Phân mảnh hỗn hợp (mixed fragmentation)**

Các loại phân mảnh dữ liệu

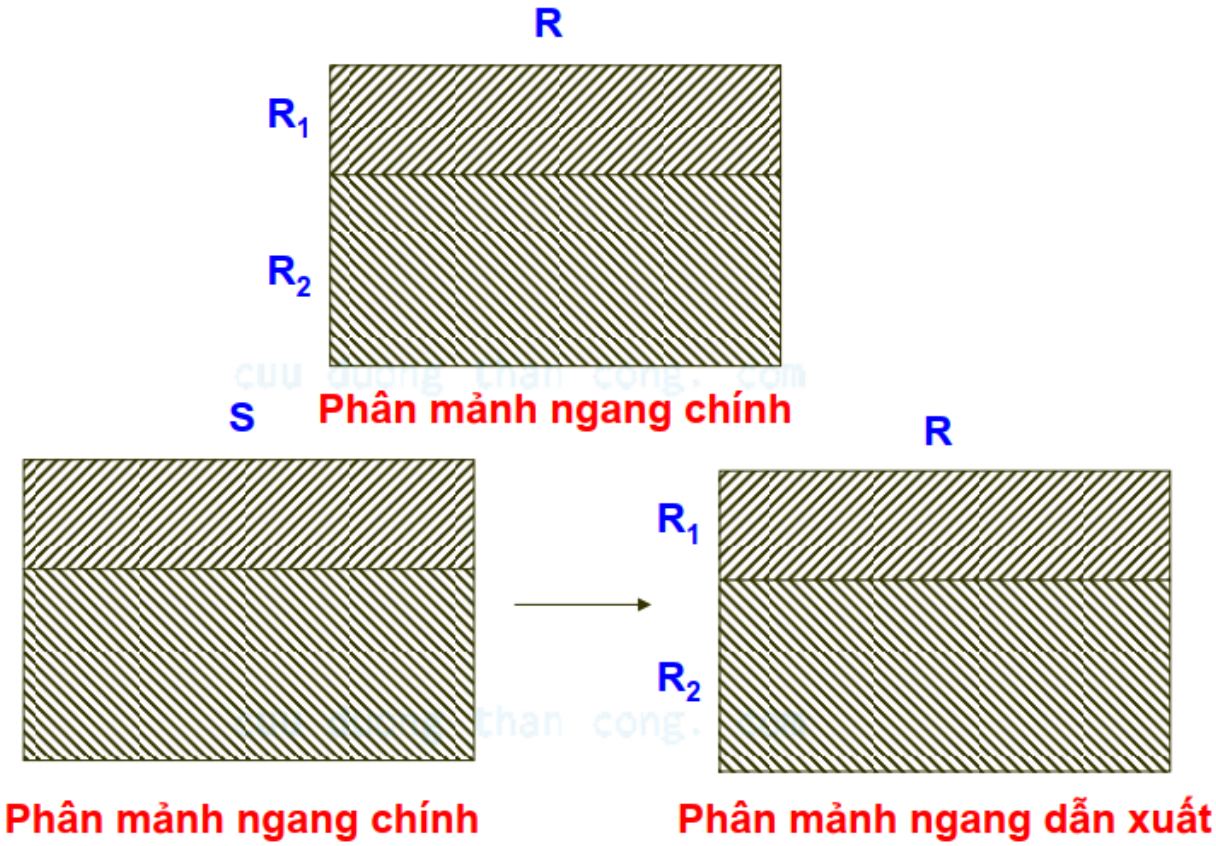
❖ Phân mảnh ngang (*Horizontal fragmentation – HF)*

- **Phân mảnh ngang** một quan hệ tổng thể n-bộ R là tách R thành các quan hệ con **R1, R2, ..., Rk** bằng **phép chọn hoặc kết**, sao cho quan hệ R có thể được khôi phục lại từ các quan hệ con này bằng phép hợp: **R = R1 ∪ R2 ∪ ... ∪ Rk**
- **Phân mảnh ngang chính *(primary HF)* :** Một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ đó (phép chọn)
- **Phân mảnh ngang dẫn xuất *(derived HF)*:** một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ khác (phép kết)

Vị từ : là điều kiện dùng để xác định một mảnh chứa dữ liệu gì

Các loại phân mảnh dữ liệu

- **Phân mảnh ngang (*Horizontal fragmentation – HF)***



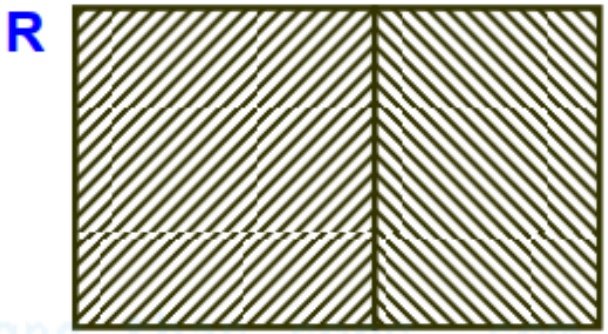
❖ Phân mảnh dọc (*Vertical fragmentation – VF)*

- Phân mảnh dọc một quan hệ tổng thể n-bộ R là tách R thành các quan hệ con n-bộ R1, R2, ..., Rk bằng **phép chiếu**, sao cho quan hệ R có thể được khôi phục lại từ các quan hệ con này bằng phép kết:

$$R = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_k$$

Các loại phân mảnh dữ liệu

❖ Phân mảnh dọc (*Vertical fragmentation – VF)*

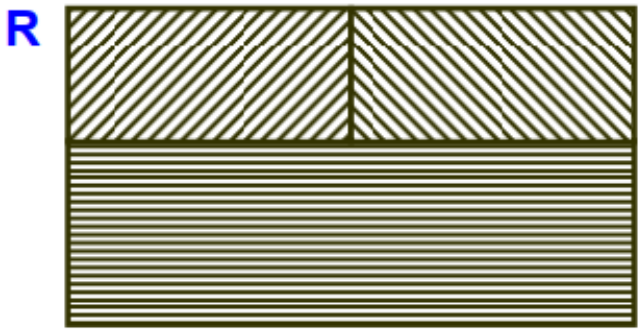


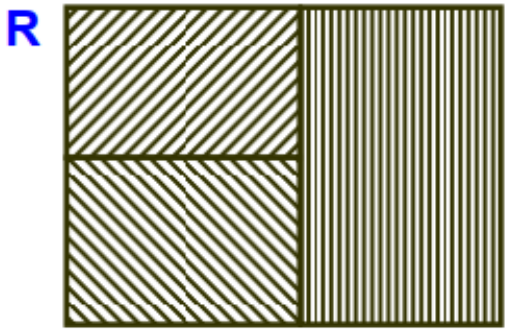
Các loại phân mảnh dữ liệu

- **Phân mảnh hỗn hợp (mixed fragmentation):** Kết hợp cả phân mảnh ngang và phân mảnh dọc
- Tái tạo bằng phép hội và kết

Các loại phân mảnh dữ liệu

- **Phân mảnh hỗn hợp (mixed fragmentation)**





Các điều kiện đúng đắn

- Quan hệ R được phân rã thành các mảnh $R_1, R_2, \ldots, R_n$
- Khi phân mảnh phải thỏa mãn các quy tắc sau để đảm bảo CSDL không bị thay đổi ngữ nghĩa trong quá trình phân mảnh
  - **Điều kiện đầy đủ (completeness condition)**
  - **Điều kiện tái tạo (reconstruction condition)**
  - **Điều kiện tách biệt (disjointness condition)**

Các điều kiện đúng đắn

- **Điều kiện đầy đủ (completeness condition)**
- Mỗi mục dữ liệu trong R phải có trong một hoặc nhiều mảnh R_i
- **Phân mảnh ngang**: mục dữ liệu là 1 bộ (tuple)

  $\forall u \in R, \exists i \in [1, n]: u \in Ri$

- **Phân mảnh dọc:** mục dữ liệu là một thuộc tính

  $\forall A \in Attr(R), \exists i \in [1, n]: A \in Attr(Ri)$ với Attr(R) là tập thuộc tính của R

Các điều kiện đúng đắn

- **Điều kiện tái tạo (reconstruction condition)**
- Luôn luôn có thể xác định một phép toán quan hệ ∇ sao cho (phép toán quan hệ ∇ tùy vào loại phân mảnh):

  $R = \nabla R_i, \forall R_i \in F_R$ với $F_R = \{R_1, R_2, \ldots, R_n\}$
- **Phân mảnh ngang:**

  $R = R_1 \cup R_2 \cup \ldots \cup R_n$
- **Phân mảnh dọc:**

  $R = R_1 \rhd\lhd R_2 \rhd\lhd \ldots \rhd\lhd R_n$
- Đảm bảo rằng các ràng buộc về mặt dữ liệu sẽ được bảo toàn

Các điều kiện đúng đắn

❖ **Điều kiện tách biệt (disjointness condition)**

Nếu mục dữ liệu d_i có trong R_i thì nó không có trong bất kỳ mảnh R_k khác ($k \neq i$).

+ Phân mảnh ngang:

$\forall i \neq k$ và $i, k \in [1, n]$: $R_i \cap R_k = \emptyset$

hoặc $\forall u \in R_i$, $\forall i \neq k$ và $i, k \in [1, n]$: $u \notin R_k$

Tiêu chuẩn này đảm bảo các mảnh ngang sẽ tách biệt. Nếu quan hệ được

+ Phân mảnh dọc: thường vi phạm

vì các thuộc tính khoá chính phải được lặp lại trong mỗi mảnh

- **Example.DDB**
- **Lược đồ toàn cục:**

EMP (EMPNUM, NAME, SAL, TAX, MGRNUM,DEPTNUM)

DEPT (DEPTNUM, NAME, AREA, MGRNUM)

SUPPLIER (SNUM, NAME, CITY)

SUPPLY (SNUM, PNUM, DEPTNUM, QUAN)

Phân mảnh ngang chính (primary horizontal fragmentation)

- Là sự phân chia các bộ của một quan hệ toàn cục thành các tập hợp con dựa vào các thuộc tính của quan hệ này
- Mỗi tập hợp con được gọi là mảnh ngang (horizontal fragment)
- Mỗi mảnh ngang được tạo bởi một phép chọn trên quan hệ toàn cục
- **Vị từ định tính (qualification):** là các vị từ (predicate) được sử dụng trong phép chọn để xác định một mảnh, hoặc còn được gọi là điều kiện chọn

Phân mảnh ngang chính

Phân mảnh ngang chính (primary horizontal fragmentation)

Ví dụ:

- **Quan hệ toàn cục:**

  SUPPLIER (SNUM, NAME, CITY)

- **Các mảnh ngang:**

  $SUPPLIER1 = \sigma_{CITY = 'SF'} SUPPLIER$

  $SUPPLIER2 = \sigma_{CITY = 'LA'} SUPPLIER$

- **Các vị từ định tính:**

  Q1: CITY = 'SF'

  Q2: CITY = 'LA'

Phân mảnh ngang chính

Phân mảnh ngang chính (primary horizontal fragmentation)

Xét các điều kiện đúng đắn:

- **Điều kiện đầy đủ:**

'SF' và 'LA' chỉ là các giá trị có thể có của thuộc tính CITY, nếu không thì sẽ không biết được các bộ ứng với các giá trị khác của CITY thuộc về mảnh nào

- **Điều kiện tái tạo:**

tái tạo mối quan hệ SUPPLIER thông qua phép hợp

SUPPLIER = SUPPLIER1 ∪ SUPPLIER2

- **Điều kiện tách biệt:**

được thỏa mãn vì SNUM là khóa của SUPPLIER

Phân mảnh ngang dẫn xuất

Phân mảnh ngang dẫn xuất: (derived horizontal fragmentation)

- Là sự phân chia các bộ của một quan hệ toàn cục thành các tập hợp con dựa vào phân mảnh ngang của một quan hệ khác (được gọi là quan hệ chủ).
- **Vị từ định tính của mảnh ngang dẫn xuất** bao gồm điều kiện kết và vị từ định tính của mảnh ngang chủ tương ứng

Phân mảnh ngang dẫn xuất

Ví dụ:

- **Quan hệ toàn cục:**

  SUPPLY (SNUM, PNUM, DEPTNUM, QUAN)

  SUPPLIER (SNUM, NAME, CITY)

Phân mảnh quan hệ SUPPLY sao cho một mảnh chứa các bộ thuộc các nhà cung cấp ở một thành phố cho trước

→ sử dụng phép nửa kết

! Phép nửa kết: R ⋉ S (chính là phép kết tự nhiên của R và S, QH kết quả sẽ chứa tập thuộc tính của R)

Phân mảnh ngang dẫn xuất

Các mảnh ngang dẫn xuất: *chọn từ SUPPLY các bộ thỏa mãn điều kiện kết giữa SUPPLIER1 hoặc SUPPLIER2 và SUPPLY*

$$SUPPLY1 = SUPPLY \triangleright\!\!< _{SNUM = SNUM} SUPPLIER1$$

$$SUPPLY2 = SUPPLY \triangleright\!\!< _{SNUM = SNUM} SUPPLIER2$$

Các vị từ định tính:

q1: SUPPLY.SNUM = SUPPLIER.SNUM AND SUPPLIER.CITY = ‘SF’

q2: SUPPLY.SNUM = SUPPLIER.SNUM AND SUPPLIER.CITY = ‘LA’

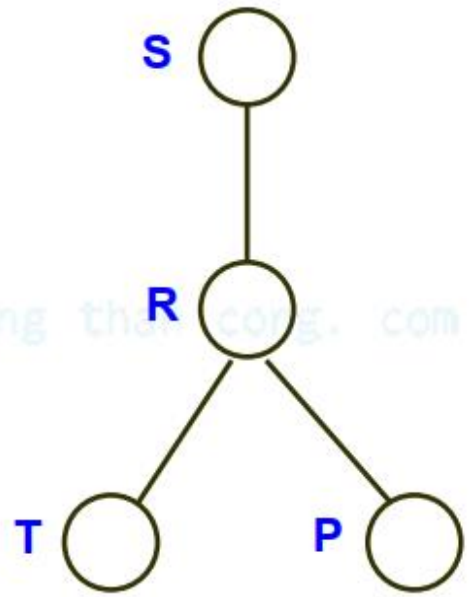
Phát biểu như sau: *đối với một bộ bất kỳ của SUPPLY1(SUPPLY2) tồn tại một nhà cung cấp ở SF hoặc LA có cùng mã nhà cung cấp (SNUM)*

Phân mảnh ngang dẫn xuất: (derived horizontal fragmentation)

- **Điều kiện đúng đắn:**
  - **Điều kiện đầy đủ:** tất cả các mã nhà cung cấp trong quan hệ SUPPLY đều phải có trong quan hệ SUPPLIER (được gọi là ràng buộc toàn vẹn tham chiếu)
  - **Điều kiện tái tạo:** phép hợp các mảnh SUPPLY1 và SUPPLY2
  - **Điều kiện tách biệt:** nếu 1 bộ của quan hệ SUPPLY không tương ứng với hai bộ của quan hệ SUPPLIER → thỏa vì các mã nhà cung cấp SNUM là các khóa duy nhất của SUPPLIER

Phân mảnh ngang dẫn xuất

- **Cây phân mảnh ngang dẫn xuất:** nếu quan hệ toàn cục R được phân mảnh ngang dẫn xuất theo quan hệ toàn cục S thì cũng có thể có hai quan hệ toàn cục T và P được phân mảnh ngang dẫn xuất theo R → biểu diễn bằng cây phân mảnh ngang dẫn xuất



Phân mảnh dọc: (vertical fragmentation)

- Là sự phân chia tập thuộc tính của một quan hệ toàn cục thành các tập thuộc tính con
- Các mảnh dọc (vertical fragment) có được bằng cách chiếu quan hệ toàn cục trên mỗi tập thuộc tính con.

Phân mảnh dọc (vertical fragmentation)

- **Sự phân mảnh này sẽ đúng đắn nếu:**
  - Mỗi thuộc tính được ánh xạ vào ít nhất vào một thuộc tính của các phân mảnh;
  - Nó phải có khả năng tái thiết lại quan hệ nguyên thủy bằng cách **kết nối** các phân mảnh lại với nhau. Để có khả năng tái thiết lại quan hệ nguyên thủy thì **mỗi phân mảnh dọc phải chứa khóa chính** của quan hệ nguyên thủy đó

Phân mảnh dọc (vertical fragmentation)

- **Quan hệ toàn cục:**

  EMP (EMPNUM, NAME, SAL, TAX, MGRNUM, DEPTNUM)

- **Phân mảnh dọc không dư thừa:**

  $EMP1 = \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$

  $EMP2 = \Pi_{EMPNUM, SAL, TAX} EMP$

- **Điều kiện tái tạo và đầy đủ:**

  $EMP = EMP1 \rhd\lhd_{EMPNUM = EMPNUM} EMP2$

Phân mảnh dọc (vertical fragmentation)

- **Quan hệ toàn cục:**

  EMP (EMPNUM, NAME, SAL, TAX, MGRNUM, DEPTNUM)

- **Phân mảnh dọc dư thừa:**

  EMP1 = $\Pi_{\text{EMPNUM, NAME, MGRNUM, DEPTNUM}}$ EMP

  EMP2 = $\Pi_{\text{EMPNUM, NAME, SAL, TAX}}$ EMP

- **Điều kiện Tái tạo và đầy đủ**

  EMP = EMP1 $\rhd\lhd_{\text{EMPNUM = EMPNUM}}$ ($\Pi_{\text{EMPNUM, SAL, TAX}}$ EMP2)

Phân mảnh dọc (vertical fragmentation)

- ***Lưu ý:*** *Điều kiện tách biệt trong phân mảnh dọc không quan trọng như trong phân mảnh ngang, vì có khóa được nhân bản trong tất cả các mảnh để tái tạo quan hệ toàn cục*
- Thường sử dụng phép chiếu
- Mỗi phân mảnh đều chứa khoá của quan hệ

Phân mảnh hỗn hợp

- Phân mảnh hỗn hợp - kết hợp của cả hai kiểu phân mảnh ngang và phân mảnh dọc.
- **Hai phương pháp phân mảnh kết hợp**
  - Phân mảnh ngang trước phân mảnh dọc
  - Phân mảnh dọc trước phân mảnh ngang

Phân mảnh hỗn hợp

- **Quan hệ toàn cục:**

  EMP (EMPNUM, NAME, SAL, TAX, MGRNUM, DEPTNUM)

- **Phân mảnh hỗn hợp:**

  $EMP1 = \sigma_{DEPTNUM \leq 10} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$

  $EMP2 = \sigma_{10 < DEPTNUM \leq 20} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$

  $EMP3 = \sigma_{DEPTNUM > 20} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$

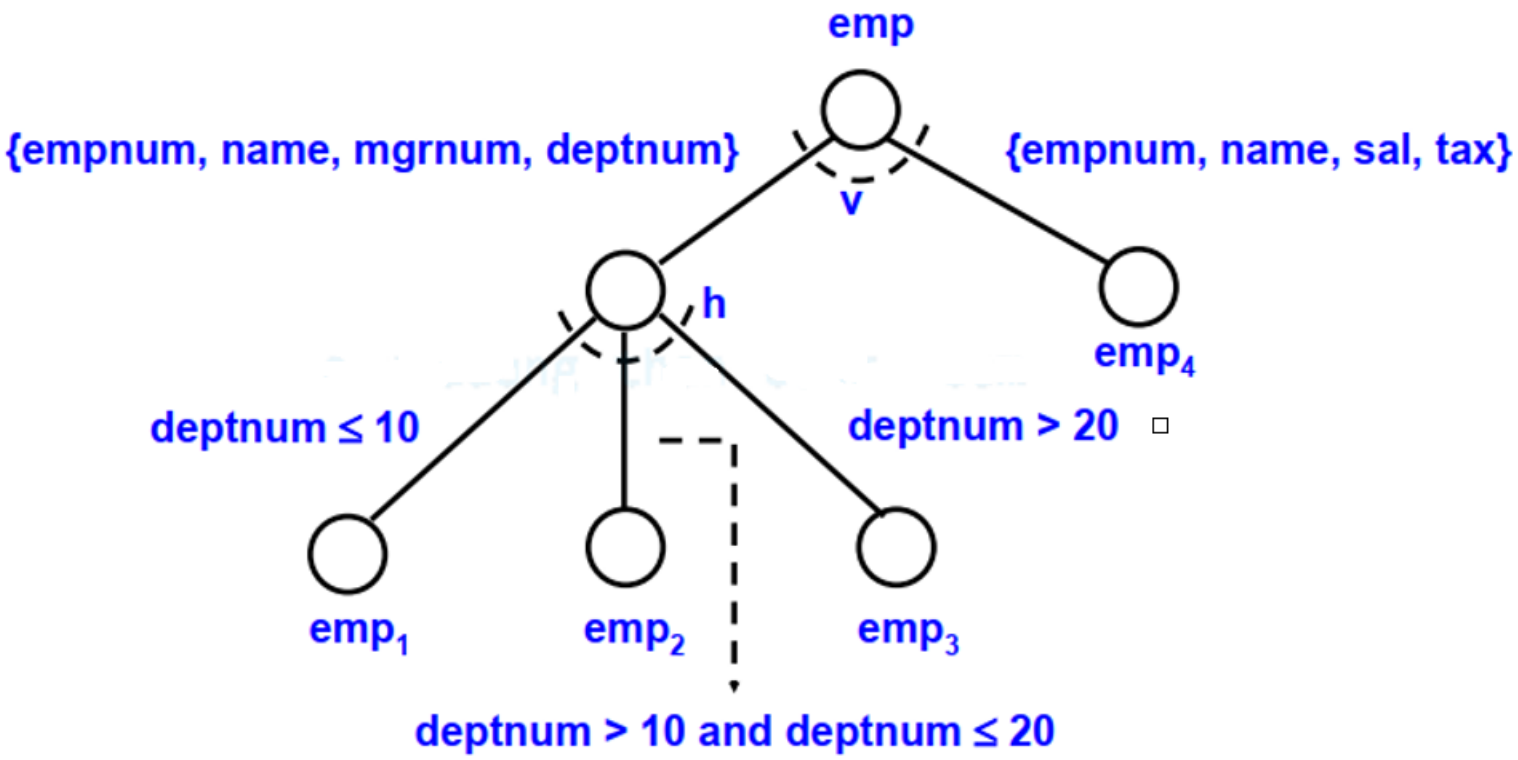
  $EMP4 = \Pi_{EMPNUM, NAME, SAL, TAX} EMP$

- **Tái tạo:**

  $EMP = (EMP1 \cup EMP2 \cup EMP3) \rhd\lhd_{EMPNUM = EMPNUM} (\Pi_{EMPNUM, SAL, TAX} EMP4)$

Phân mảnh hỗn hợp

- **Cây phân mảnh**



Example.DDB

- **Lược đồ toàn cục:**

  EMP (**EMPNUM**, NAME, SAL, TAX, MGRNUM,DEPTNUM)

  DEPT (**DEPTNUM**, NAME, AREA, MGRNUM)

  SUPPLIER (**SNUM**, NAME, CITY)

  SUPPLY (**SNUM**, **PNUM**, DEPTNUM, QUAN)

Example.DDB

❖ LƯỢC ĐỒ PHÂN MẢNH:

$EMP1 = \sigma_{DEPTNUM \le 10} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$
$EMP2 = \sigma_{10 < DEPTNUM \le 20} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$
$EMP3 = \sigma_{DEPTNUM > 20} \Pi_{EMPNUM, NAME, MGRNUM, DEPTNUM} EMP$
$EMP4 = \Pi_{EMPNUM, NAME, SAL, TAX} EMP$

$DEPT1 = \sigma_{DEPTNUM \le 10} DEPT$
$DEPT2 = \sigma_{10 < DEPTNUM \le 20} DEPT$
$DEPT3 = \sigma_{DEPTNUM > 20} DEPT$

$SUPPLIER1 = \sigma_{CITY = 'SF'} SUPPLIER$
$SUPPLIER2 = \sigma_{CITY = 'LA'} SUPPLIER$

$SUPPLY1 = SUPPLY \triangleright\!<_{SNUM = SNUM} SUPPLIER1$
$SUPPLY2 = SUPPLY \triangleright\!<_{SNUM = SNUM} SUPPLIER2$

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

- Để hiểu được sự trong suốt phân tán của ứng dụng chỉ đọc, chúng ta sẽ xem xét các ví dụ được minh họa bằng ngôn ngữ tựa Pascal có nhúng ngôn ngữ SQL
- Nhập xuất được thực hiện qua các thủ tục chuẩn: Read(terminal, variable), write(terminal, variable)

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

- Trong các phát biểu SQL, các biến của ngôn ngữ Pascal sẽ bắt đầu bằng ký tự “$”
- Ví dụ: Xét câu truy vấn : Tìm tên của người cung cấp khi biết mã số người cung cấp.

```
Read (terminal, $SNUM)
Select NAME into $NAME From SUPPLIER Where SNUM = $SNUM
write (terminal, $NAME)
```

Trong câu truy vấn này $NAME là biến xuất còn $SNUM là biến nhập.

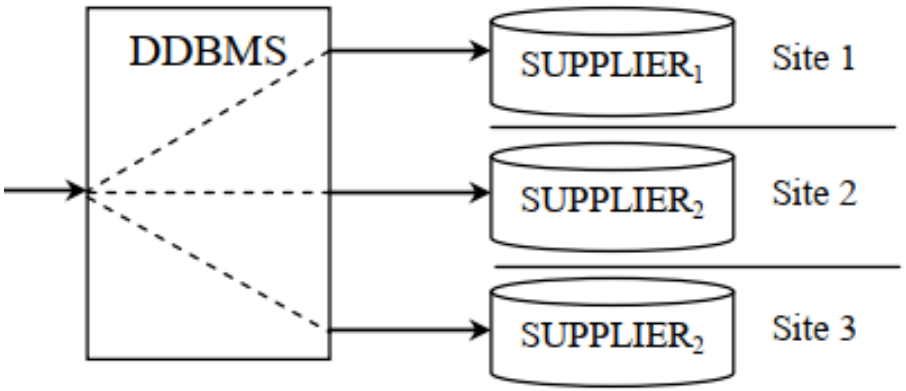
Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

- Các biến của Pascal sử dụng để liên hệ với hệ quản trị cơ sở dữ liệu phân tán bắt đầu bằng ký tự “#”.
- Ví dụ: Sau khi truy vấn một câu SQL
  - Biến luận lý #FOUND = TRUE nếu kết quả trả về khác rỗng (@@ROWCOUNT>0)
  - Biến #OK = TRUE (@@ERROR =0) nếu phép toán thực hiện đúng bởi hệ quản trị cơ sở dữ liệu

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

- **Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.**
- **Mức 1: Trong suốt phân mảnh**

```
Read (terminal, $SNUM)
    Select NAME into $NAME
    from SUPPLIER
    Where SNUM = $SNUM
if #FOUND then write (terminal, $NAME)
    else write (terminal, 'Not found');
```



Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

- **Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.**
- **Mức 1: Trong suốt phân mảnh**

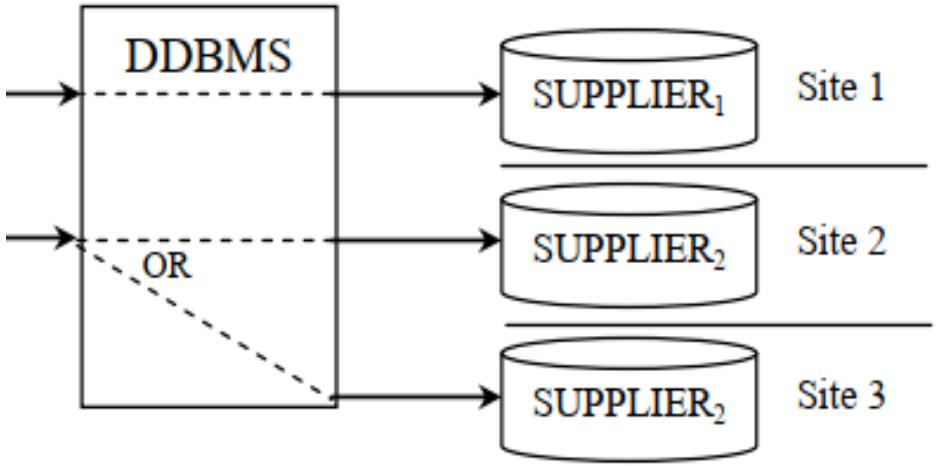
Nhận xét: chúng ta thấy câu truy vấn trên **tương tự như câu truy vấn cục bộ**, không cần chỉ ra các phân mảnh cũng như các vị trí cấp phát cho các phân mảnh đó. Khi đó người sử dụng **không hề có cảm giác là đang thao tác trên một câu truy vấn phân tán.**

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

❖ Mức 2: Trong suốt vị trí

```
Read (terminal, $SNUM)
    Select NAME into $NAME
    From SUPPLIER1
    Where SNUM = $SNUM
If not #Found then
    Select NAME into $NAME
    From SUPPLIER2
    Where SNUM = $SNUM
if #FOUND then write (terminal, $NAME)
    else write (terminal, 'Not found');
```



Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

- **Mức 2: Trong suốt vị trí**

Nhận xét: Người sử dụng **phải cung cấp tên các phân mảnh cụ thể** cho câu truy vấn nhưng không cần chỉ ra vị trí của các phân mảnh

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

- **Mức 2: Trong suốt vị trí**

  Chương trình có thể viết lại như sau khi thêm thành phố của nhà cung cấp (vị từ định tính của mảnh)

  Nhận xét: khi biết thêm thành phố của nhà cung cấp thì ứng dụng chỉ tham chiếu đến một trong hai mảnh

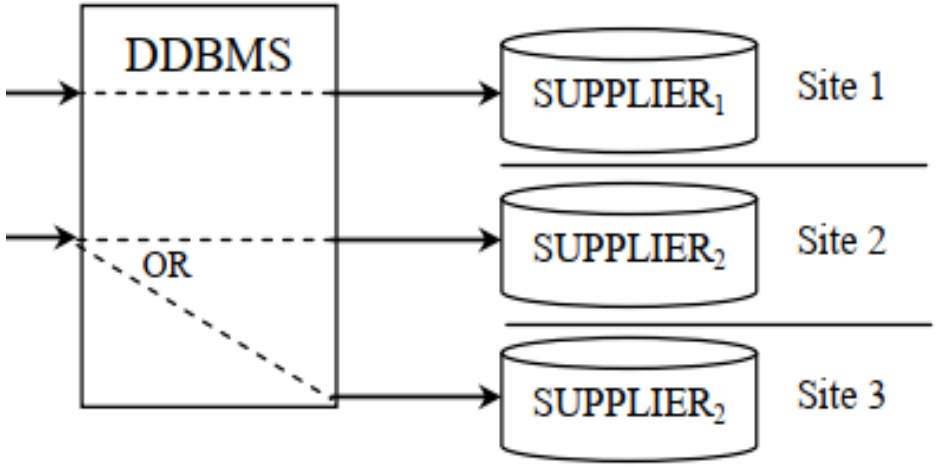
```
Read(Terminal, $SNUM)
Read(Terminal, $CITY)
Case $CITY Of
"SF":
        Select NAME into $NAME
        From SUPPLIER1
        Where SNUM = $SNUM;
"LA":
        Select NAME into $NAME
        From SUPPLIER2
        Where SNUM = $SNUM
End;
if #FOUND then write (terminal, $NAME)
else write (terminal, 'Not found');
```

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

❖ Mức 3: Trong suốt ánh xạ cục bộ

```
Read (terminal, $SNUM)
Select NAME into $NAME
From SUPPLIER1 AT SITE 1
Where SNUM = $SNUM
 If not #Found then
      Select NAME into $NAME
      From SUPPLIER2 AT SITE 2
      Where SNUM = $SNUM
if #FOUND then write (terminal, $name)
else write (terminal, 'Not found');
```



Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

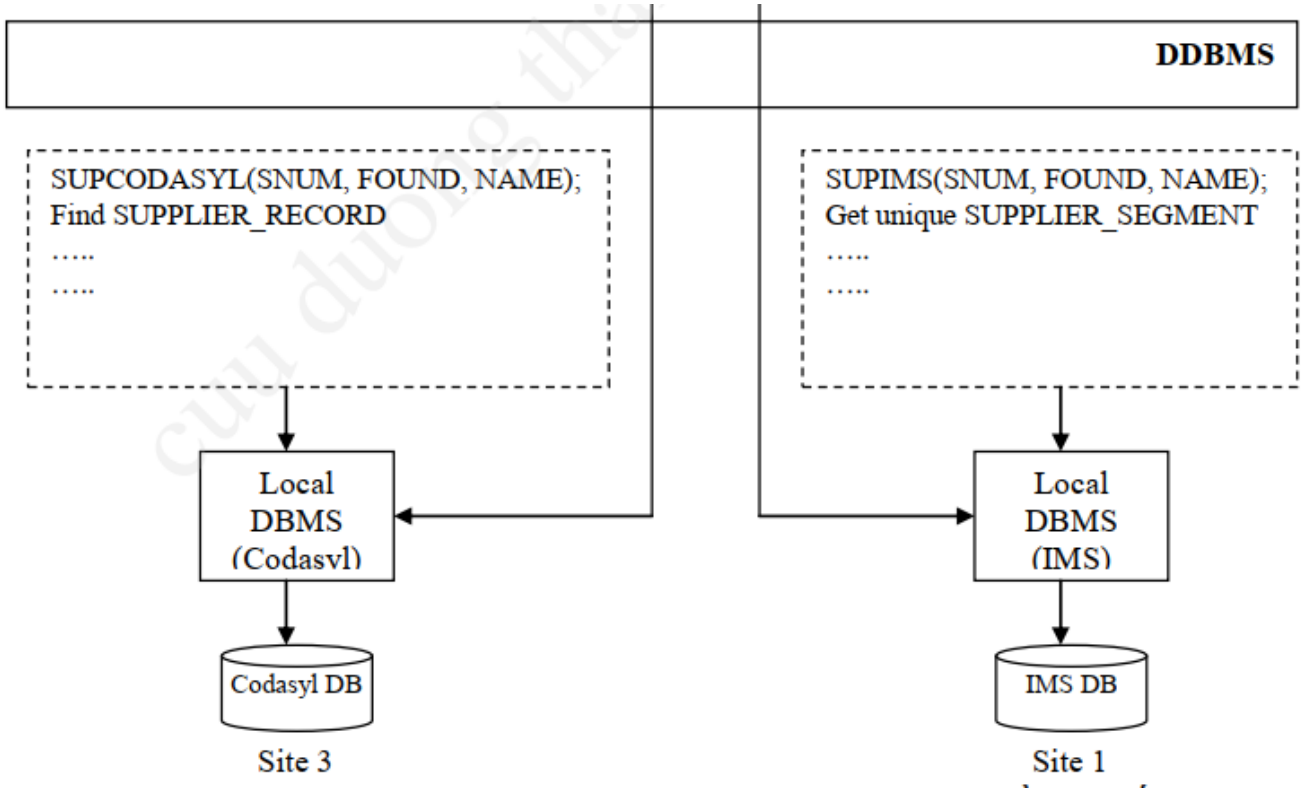
Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

- **Mức 3: Trong suốt ánh xạ cục bộ**

Nhận xét: tại mức trong suốt này người sử dụng phải cung cấp tên các phân mảnh và vị trí cấp phát của chúng

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

❖ Mức 4: Không trong suốt



Nhận xét: Tại mức thấp nhất này chúng ta cần phải viết lệnh theo hệ quản trị cơ sở dữ liệu tương ứng

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 1: Cho biết tên của nhà cung cấp có mã được nhập từ thiết bị đầu cuối.

- **Mức 4: Không trong suốt**

```
Read(Terminal,SSUPNUM)
Execute SUPIMS(SSUPNUM, $FOUND, $NAME) at site 1;
if not $FOUND then
Execute SUPCODASYL(SSUPNUM, $FOUND, $NAME) at site 3;
 if #FOUND then write (terminal, $ NAME)
      else write (terminal, 'Not found');
```

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 2:

- Cho biết tên của nhà cung cấp mà họ cung cấp mặt hàng có mã được nhập từ thiết bị đầu cuối.
- Giả sử một mặt hàng chỉ được cung cấp bởi một nhà cung cấp.

SUPPLIER (**SNUM**, NAME, CITY)

SUPPLY (**SNUM**, **PNUM**, DEPTNUM, QUAN)

$$SUPPLIER1 = \sigma_{CITY = 'SF'} SUPPLIER$$
$$SUPPLIER2 = \sigma_{CITY = 'LA'} SUPPLIER$$

$$SUPPLY1 = SUPPLY \triangleright\!<_{SNUM = SNUM} SUPPLIER1$$
$$SUPPLY2 = SUPPLY \triangleright\!<_{SNUM = SNUM} SUPPLIER2$$

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 2:

Mức 1: trong suốt phân mảnh

```
Read(Terminal, $PNUM)
Select NAME into $NAME
from SUPPLIER, SUPPLY
where SUPPLIER.SNUM = SUPPLY.SNUM
      and SUPPLY.PNUM = $PNUM
if #FOUND then write (terminal, $ NAME)
      else write (terminal, 'Not found');
```

Nhận xét: như làm việc ở quan hệ toàn cục, truy vấn có thêm một phép kết

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 2:

Nhận xét: có nhiều cách để thực hiện truy vấn, người lập trình phải xác định các chiến lược cho phù hợp (chỉ sử dụng hai phép kết thay vì bốn phép kết)

Mức 2: trong suốt vị trí

```
Read(Terminal, $PNUM)
Select NAME into $NAME
from SUPPLIER1, SUPPLY1
where SUPPLIER1.SNUM = SUPPLY1.SNUM
and SUPPLY1.PNUM = $PNUM
if not #FOUND then
        Select NAME into $NAME
         from SUPPLIER2, SUPPLY2
        where SUPPLIER2.SNUM = SUPPLY2.SNUM a
        nd SUPPLY2.PNUM = $PNUM
if #FOUND then write (terminal, $ NAME)
        else write (terminal, 'Not found');
```

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 2:

Mức 3: trong suốt ánh xạ cục bộ:

Giả sử các sơ đồ cấp phát các phân mảnh của quan hệ SUPPLY và SUPPLIER như sau:

$SUPPLIER_1$: Tại site 1

$SUPPLIER_2$: Tại site 2

$SUPPLY_1$: Tại site 3

$SUPPLY_2$: Tại site 4

Tính trong suốt phân tán dùng cho ứng dụng chỉ đọc

Ví dụ 2:

SUPPLIER1 : Tại site 1
SUPPLIER2 : Tại site 2
SUPPLY1 : Tại site 3
SUPPLY2 : Tại site 4

```
Read(Terminal, $PNUM)
Select SNUM into $SNUM
from SUPPLY1 at site 3
where SUPPLY1.PNUM = $PNUM
if #FOUND then
        begin
                send $SNUM from site 3 to site 1
                Select NAME into $NAME
                from SUPPLIER1 at site 1
                where SUPPLIER1.SNUM = $SNUM
        end
else
        begin
                Select SNUM into $SNUM
                from SUPPLY2 at site 4
                where  SUPPLY2.PNUM = $PNUM
        if #FOUND then
                begin
                        send $SNUM from site 4 to site 2
                        Select NAME into $NAME
                        from SUPPLIER2 at site 2
                        where SUPPLIER2.SNUM =
                        $SNUM
                end;
end;
if #FOUND then write (terminal, $ NAME)
        else write (terminal, 'Not found');
```

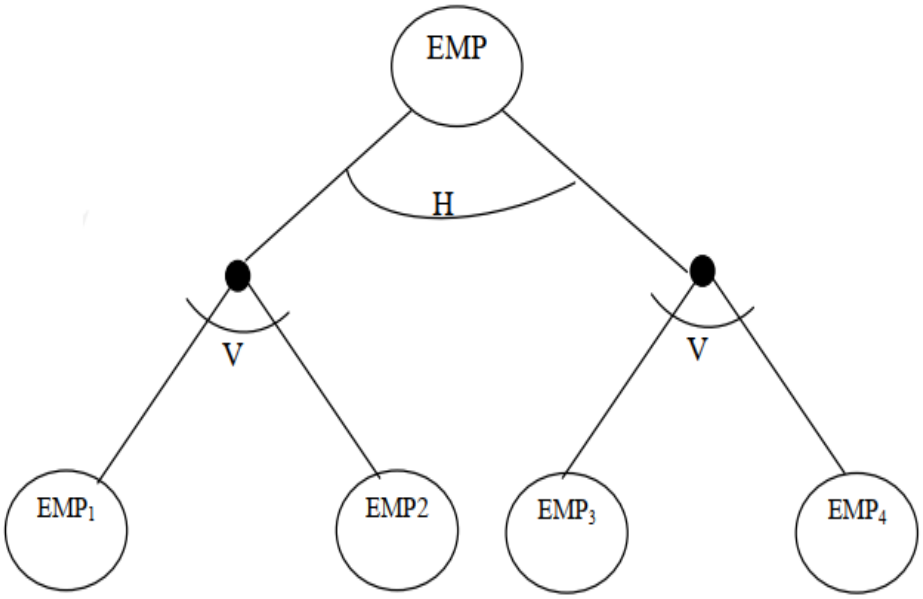
Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- Cập nhật dữ liệu (thêm, sửa, xóa) phải bảo đảm các ràng buộc toàn vẹn về khóa chính, khóa ngoại, ràng buộc nghiệp vụ ...
- Một phép cập nhật phải được thực hiện trên tất cả các bản sao của một mục dữ liệu trong khi phép tìm kiếm chỉ cần thực hiện trên một bản sao**. (read one, write all)**
- Trong việc hỗ trợ sự trong suốt phân tán cho các ứng dụng cập nhật có một vấn đề khác phức tạp hơn việc cập nhật tất cả các bản sao của một mục dữ liệu. Đó là vấn đề di chuyển dữ liệu sau khi cập nhật

Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- **Xét ví dụ sau:** Sửa dữ liệu của nhân viên có mã 100: mã phòng 3 thành mã phòng 15.

EMP1 được đặt tại nơi 1 và 5.
EMP2 được đặt tại nơi 2 và 6.
EMP3 được đặt tại nơi 3 và 7.
EMP4 được đặt tại nơi 4 và 8.



$$EMP1 = \Pi_{EMPNUM,\ NAME,\ SAL,\ TAX}\ \sigma_{DEPTNUM <= 10}\ EMP$$

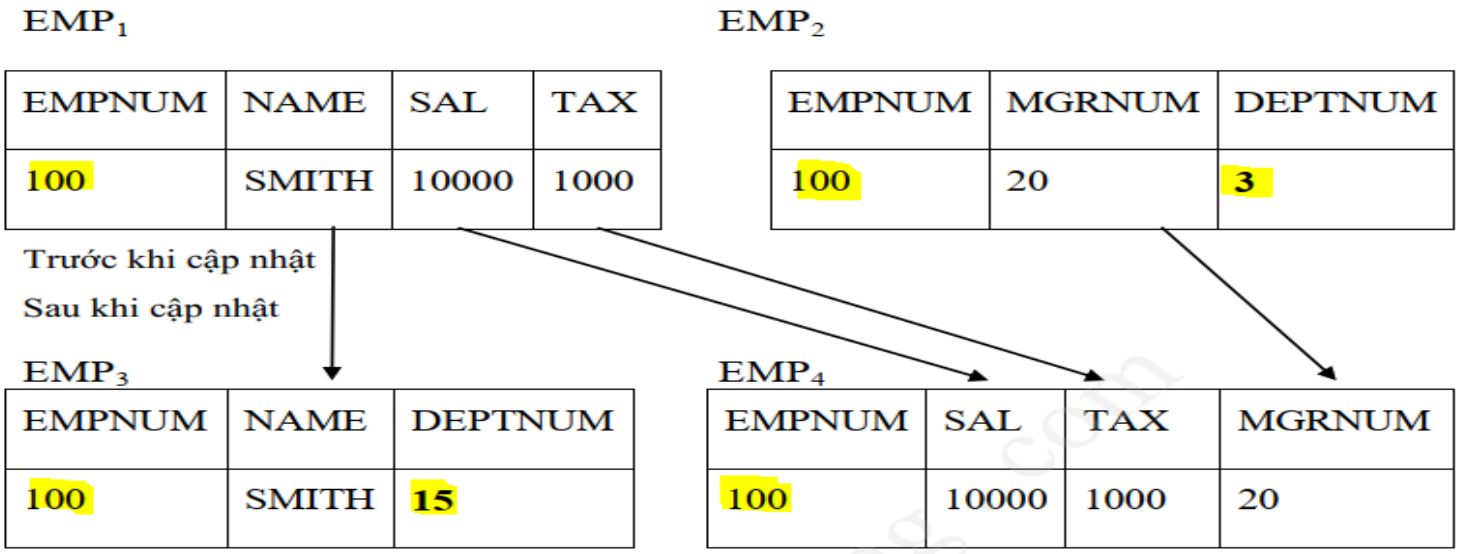
$$EMP2 = \Pi_{EMPNUM,\ MGRNUM,\ DEPTNUM}\ \sigma_{DEPTNUM <= 10}\ EMP$$

$$EMP3 = \Pi_{EMPNUM,\ NAME,\ DEPTNUM}\ \sigma_{DEPTNUM > 10}\ EMP$$

$$EMP4 = \Pi_{EMPNUM,\ SAL,\ TAX,\ MGRNUM}\ \sigma_{DEPTNUM > 10}\ EMP$$

Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- **Xét ví dụ sau:** Sửa dữ liệu của nhân viên có mã 100: mã phòng 3 thành mã phòng 15.



Sự ảnh hưởng của việc thay đổi giá trị của **DEPTNUM** của bộ có **EMPNUM=100 từ 3 thành 15**. Sự cập nhật của bộ này chỉ thuộc về cây con trái, sau khi cập nhật nó trở thành một phần cây con phải. Chúng ta nhận thấy không chỉ dữ liệu được chuyển giữa các mảnh mà bộ này cũng được tổ hợp lại theo một cách khác

Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- **Mức 1: Trong suốt phân mảnh**

Tại mức này, lập trình viên phải làm việc với các quan hệ toàn cục

```
UPDATE EMP
Set DEPTNUM=15
Where EMPNUM=100
```

Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- **Mức 2: Trong suốt vị trí**

Tại mức này, lập trình viên phải làm việc với các phân mảnh một cách tường minh

```
Select NAME, SAL, TAX
into $NAME, $SAL, $TAX
from EMP1
where  EMPNUM=100;
if #FOUND then
```

```
begin
        Select MGRNUM into $MGRNUM
        from EMP2
         where EMPNUM=100;
        Insert into EMP3 (EMPNUM, NAME, DEPTNUM)
        Values (100, $NAME, 15);
         Insert into EMP4 (EMPNUM, SAL, TAX, MGRNUM)
        Values (100, $SAL, $TAX, $MGRNUM);
        Delete EMP1 where EMPNUM = 100;
        Delete EMP2 where EMPNUM = 100;
End
```

Tính trong suốt phân mảnh dùng cho các ứng dụng cập nhật

- **Mức 3: trong suốt ánh xạ cục bộ**

Tại mức này ứng dụng phải giải quyết vị trí của các phân mảnh một cách tường minh.

Select NAME, SAL, TAX into $NAME, $SAL, $TAX from EMP1 at site 1 where EMPNUM=100;

Select MGRNUM into $MGRNUM from EMP2 at site 2 where EMPNUM=100;

Insert into EMP3 (EMPNUM, NAME, DEPTNUM) at site 3 Values (100, $NAME, 15);

Insert into EMP3 (EMPNUM, NAME, DEPTNUM) at site 7 Values (100, $NAME, 15);

Insert into EMP4 (EMPNUM, SAL, TAX, MGRNUM) at site 4 Values (100, $SAL, $TAX, $MGRNUM);

Insert into EMP4 (EMPNUM, SAL, TAX, MGRNUM) at site 8 Values (100, $SAL, $TAX, $MGRNUM);

Delete EMP1 at site 1 where EMPNUM = 100;

Delete EMP1 at site 5 where EMPNUM = 100;

Delete EMP2 at site 2 where EMPNUM = 100;

Delete EMP2 at site 6 where EMPNUM = 100;

EMP1 được đặt tại nơi 1 và 5.
EMP2 được đặt tại nơi 2 và 6.
EMP3 được đặt tại nơi 3 và 7.
EMP4 được đặt tại nơi 4 và 8.

Các tác vụ cơ bản truy xuất CSDLPT

- **Trả về kết quả một giá trị đơn** (như trong các ví dụ trước) $NAME, $PNUM
- **Trả về kết quả là các quan hệ**

  Ví dụ: Cho câu SQL sau:

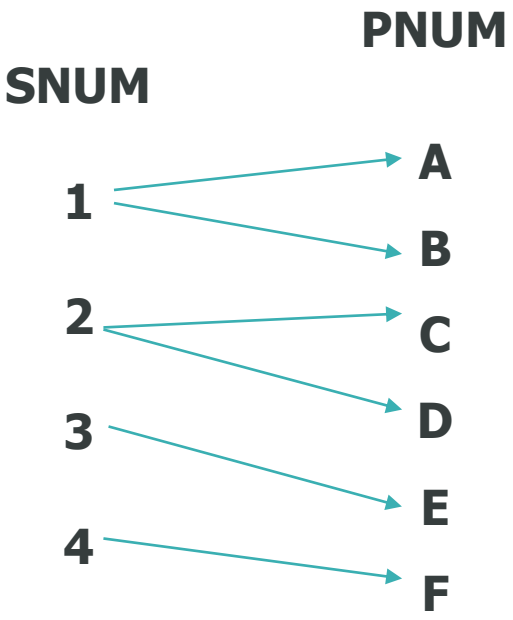
Select EMPNUM, NAME from EMP → NHIỀU DÒNG KẾT QUẢ

Select EMPNUM, NAME INTO **$EMP_REL($EMPNUM, $NAME)** from EMP

Các tác vụ cơ bản truy xuất CSDLPT

- **Ví dụ: xét chương trình ứng dụng lấy tất cả các mặt hàng được cung cấp bởi một tập hợp các nhà cung cấp**

SUPPLY (**SNUM**, **PNUM**, DEPTNUM, QUAN)



Các tác vụ cơ bản truy xuất CSDLPT

- **Cách 1: Cơ sở dữ liệu được truy xuất với mỗi giá trị $SNUM**

```
Repeat
        read(terminal, $SNUM);
        select PNUM into $PNUM_REL($PNUM)
        from SUPPLY where SNUM = $SNUM;
        repeat
                read($PNUM_REL, $PNUM)
                write(terminal, $PNUM)
        until END-OF-$PNUM_REL
until END-OF-TERMINAL-INPUT
```

Các tác vụ cơ bản truy xuất CSDLPT

- **Cách 2: Cơ sở dữ liệu được truy xuất sau khi tất cả giá trị của $SNUM được nhập**

Repeat

read(terminal, $SNUM);

write($SNUM_REL($SNUM), $SNUM)

Until END-OF-TERMINAL-INPUT

select PNUM into $PNUM_REL($PNUM)

from SUPPLY, $SNUM_REL

where SUPPLY.SNUM = $SNUM_REL.$SNUM;

Repeat

read($PNUM_REL, $PNUM)

write(terminal, $PNUM)

Until END-OF-$PNUM_REL

Các tác vụ cơ bản truy xuất CSDLPT

- **Cách 3: Cơ sở dữ liệu được truy xuất trước khi nhập giá trị $SNUM**

Select PNUM, SNUM into **$TEMP_REL**($TEMP_PNUM, $TEMP_SNUM)

from SUPPLY;

Repeat

read(terminal, $SNUM);

select $TEMP_PNUM into $TEMP2_REL($TEMP2_PNUM)

from $TEMP_REL where $TEMP_SNUM = $SNUM;

Repeat

read($TEMP2_REL, $TEMP2_PNUM);

write(terminal, $TEMP2_PNUM);

until END-OF-$TEMP2_PNUM

Until END-OF-TERMINAL-INPUT

Các tác vụ cơ bản truy xuất CSDLPT

- **Nhận xét:**
- **Cách 1:** truy xuất CSDL mỗi lần lặp (theo số lần của user nhập $SNUM vào)
- **Cách 2:** thu thập tất cả các mã $SNUM và truy xuất CSDL 1 lần để lấy dữ liệu (tuy nhiên phải thực hiện phép kết)
- **Cách 3:** truy xuất 1 lần lấy all data cần thiết lưu vào temp, thích hợp nếu bộ SUPPLY lấy ra nhiều, không đòi hỏi dữ liệu tức thời

Biểu thức con chung (common subexpression)

- Ví dụ: lấy tên các nhân viên làm việc trong một phòng ban thuộc một vùng cho trước và các mã mặt hàng của các mặt hàng được lưu trữ tại các phòng ban thuộc cùng 1 vùng
  - **Biểu thức con chung:** tập hợp các phòng ban thuộc một vùng cho trước (do phải xác định 2 lần)

Biểu thức con chung (common subexpression)

Read(terminal, $AREA);

…

Insert into $NAME_REL($NAME)

Select NAME

From EMP, **DEPT**

Where EMP.DEPTNUM=DEPT.DEPTNUM AND AREA = $AREA

…

Insert into $NAME_REL($PNUM)

Select PNUM

From SUPPLY, **DEPT**

Where SUPPLY.DEPTNUM=DEPT.DEPTNUM AND AREA = $AREA

…

Biểu thức con chung (common subexpression)

Dùng biểu thức con chung

Read(terminal, $AREA);

…

Insert into $NUM_REL($DEPTNUM)

Select DEPTNUM

From DEPT

Where AREA=$AREA

…

Insert into $NAME_REL($NAME)

Select NAME

From EMP, $NUM_REL

Where EMP.DEPTNUM=$NUM_REL.$DEPTNUM

….

Insert into $PNUM_REL($NAME)

Select PNUM

From SUPPLY, $NUM_REL

Where SUPPLY.DEPTNUM=$NUM_REL.$DEPTNUM

Ràng buộc toàn vẹn trong CSDL PT

- **Ràng bộc toàn vẹn (integrity constraint):** nghĩa là các quy định về các giá trị cho phép của dữ liệu. Khi một cập nhật làm vi phạm một ràng buộc toàn vẹn thì ứng dụng sẽ không chấp nhận → tính đúng đắn của dữ liệu vẫn được bảo đảm
- **Ràng buộc tham chiếu (referential constraint):** đòi hỏi tất cả các giá trị của một thuộc tính cho trước của một quan hệ cũng tồn tại trong một quan hệ khác nào đó

Ràng buộc toàn vẹn trong CSDL PT

- **Ví dụ:** quan hệ SUPPLY được phân mảnh ngang dẫn xuất theo quan hệ SUPPLIER bởi phép nửa kết trên thuộc tính SNUM, do đó tất cả các giá trị của SNUM trong SUPPLY phải có trong SNUM của SUPPLIER
- Trường hợp muốn loại bỏ một bộ trong quan hệ SUPPLIER

  Delete from SUPPLIER

  Where SNUM=$SNUM
- Có thể vi phạm ràng buộc tham chiếu

Ràng buộc toàn vẹn trong CSDL PT

- Do đó sửa lại như sau:

```
Select 1

From SUPPLY

Where SNUM=$SNUM

It not #FOUND then

        Delete from SUPPLIER

        Where SNUM=$SNUM
```