Chương 4 – Tính trong suốt phân tán

Giảng viên: ThS. Nguyễn Hồ Duy Tri
tringuyen@uit.edu.vn

Tính trong suốt phân tán

- **Tính trong suốt của một hệ phân tán** được hiểu như là việc **che khuất (giấu) đi** những **chi tiết phụ thuộc hệ thống** mà không thích hợp đối với người dùng trong mọi hoàn cảnh và **tạo ra một môi trường thuần nhất** cho người dùng.
- Sự che khuất thông tin phụ thuộc hệ thống khỏi người dùng dựa trên việc **cân bằng giữa tính đơn giản và tính hiệu quả**, hai tính chất này là **xung đột nhau**.
- Hệ phân tán tốt là **cố gắng đạt được tính trong suốt cao nhất** có thể được.

Tính trong suốt phân tán

- Tính trong suốt thể hiện trong nhiều khía cạnh, dưới đây là một số khía cạnh điển hình nhất:
  - **Trong suốt truy cập** (*Access transparency*): che giấu sự khác biệt về cách biểu diễn và truy cập tài nguyên.
  - **Trong suốt về vị trí** (*Location transparency*): che giấu vị trí của tài nguyên.
  - **Trong suốt di trú** (*Migration transparency*): che giấu khả năng chuyển vị trí của tài nguyên.
  - **Trong suốt về việc định vị lại** (*Relocation transparency*): che giấu việc di chuyển của tài nguyên khi đang được sử dụng.

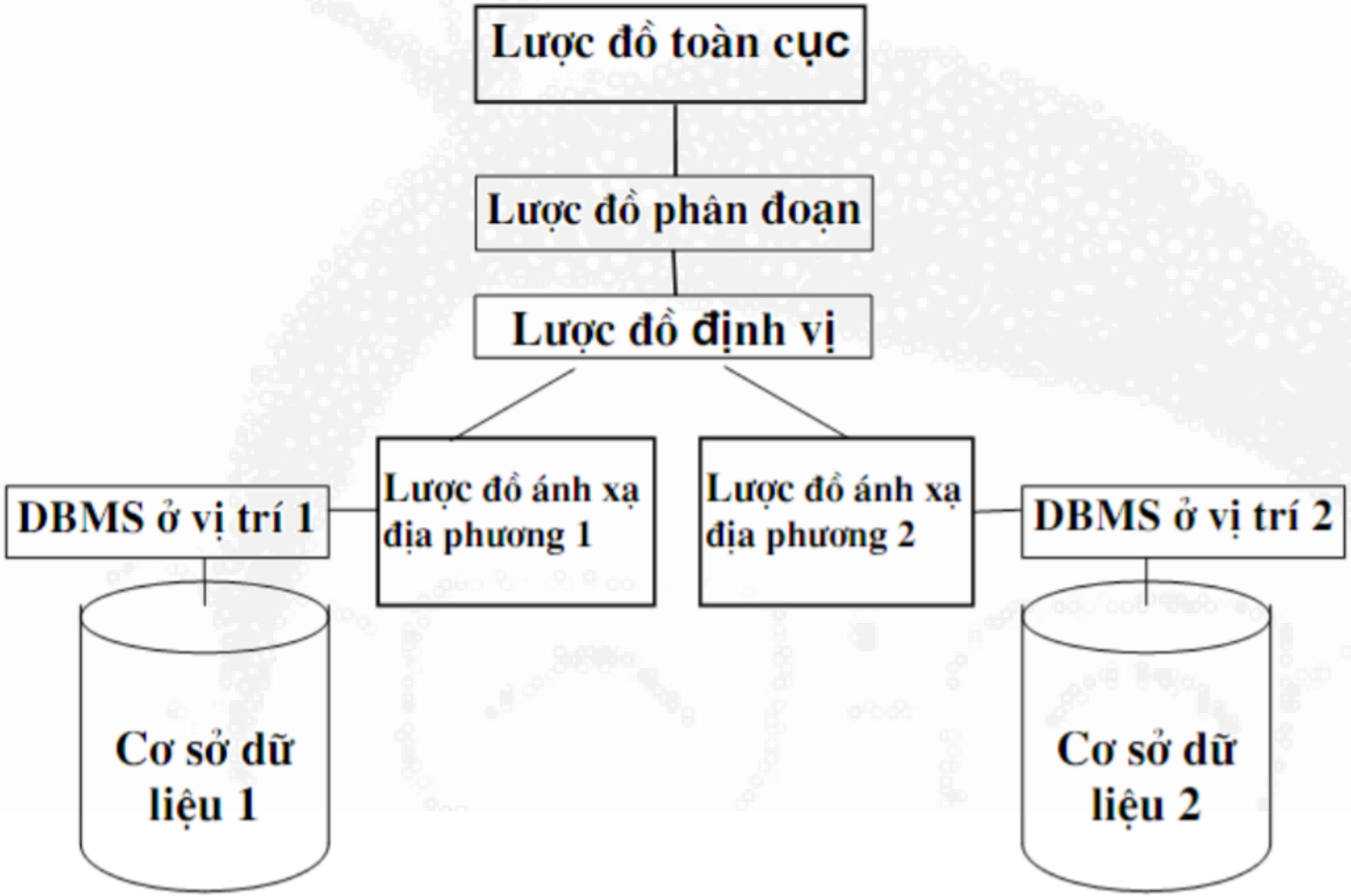
Tính trong suốt phân tán

- Tính trong suốt thể hiện trong nhiều khía cạnh, dưới đây là một số khía cạnh điển hình nhất:
  - **Trong suốt nhân bản** (*Replication transparency*): che giấu tình trạng sử dụng bản sao của tài nguyên.
  - **Trong suốt đồng thời** (*Concurrency transparency*): che giấu sự chia sẻ tài nguyên tương tranh.
  - **Trong suốt sự cố** (*Failure transparency*): che giấu lỗi hệ thống nếu có.
  - **Trong suốt khả năng di chuyển tài nguyên** (*Persistence transparency*): che giấu việc di chuyển tài nguyên từ bộ nhớ ngoài vào bộ nhớ trong và ngược lại.

Tính trong suốt phân tán

- Các mức trong suốt trong hệ phân tán:
  - Mức 0: **Không trong suốt** (*No transparency*)
  - Mức 1: **Trong suốt phân đoạn** (*Fragmentation transparency*)
  - Mức 2: **Trong suốt về vị trí** (*Location transparency*)
  - Mức 3: **Trong suốt ánh xạ địa phương** (*Local mapping transparency*)

Kiến trúc chung của cơ sở dữ liệu phân tán



Tính trong suốt phân tán

- **Trong suốt phân đoạn** (*Fragmentation transparency*):
  - ✓ Không nhìn thấy các đoạn.
  - ✓ Nhìn thấy các quan hệ toàn cục (*Global relation*)
  - ✓ Lược đồ toàn cục (*Global schema*).
- **Trong suốt vị trí** (*Location transparency*):
  - ✓ Không nhìn thấy các quan hệ cục bộ.
  - ✓ Nhìn thấy các đoạn (*Fragment*).
  - ✓ Lược đồ phân đoạn (*Fragmentation schema*)

Tính trong suốt phân tán

- **Trong suốt ánh xạ địa phương/cục bộ** (*Local mapping transparency*):
  - ✓ Nhìn thấy các quan hệ cục bộ (*Local relation*).
  - ✓ Không nhìn thấy cơ sở dữ liệu vật lý.

Tính trong suốt phân tán

Trong suốt phân đoạn (*Fragmentation transparency*):

- Khi dữ liệu đã được phân đoạn thì việc truy cập vào cơ sở dữ liệu được **thực hiện bình thường như là chưa bị phân tán** và không ảnh hưởng tới người sử dụng.
- *Ví dụ*: Xét quan hệ tổng thể NCC (ID, Tên, Tuổi) và các phân đoạn được tách ra từ nó:

  NCC1 (ID, Tên, Tuổi)
  NCC2 (ID, Tên, Tuổi)
  NCC3 (ID, Tên, Tuổi)

- Giả sử DDBMS cung cấp tính trong suốt về phân đoạn, 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.

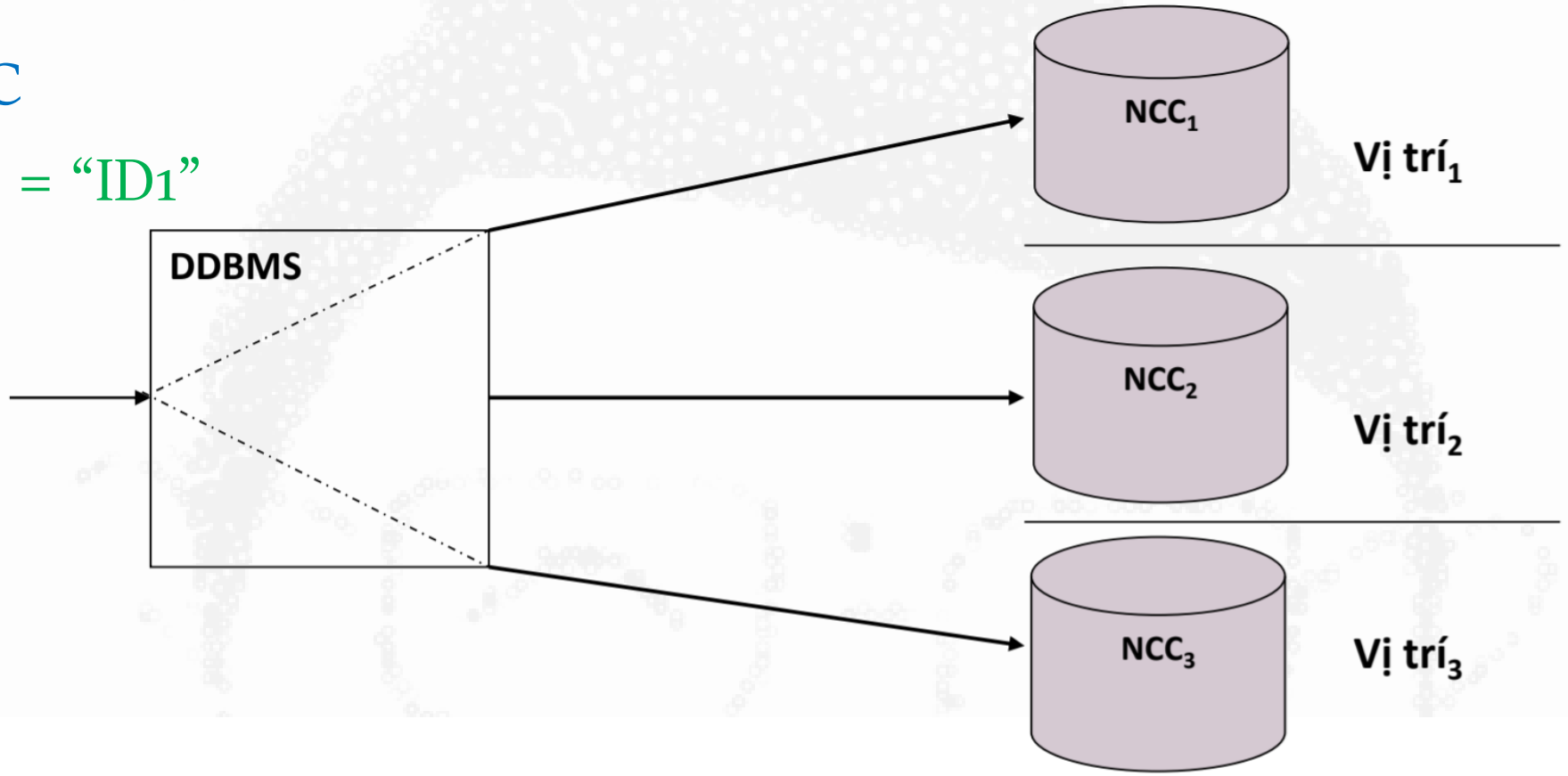
Tính trong suốt phân tán

Trong suốt phân đoạn (*Fragmentation transparency*):

select *

from NCC

where ID = "ID1"



Tính trong suốt phân tán

Trong suốt về vị trí (*Location transparency*):

- Người sử dụng **không cần biết về vị trí vật lý** của dữ liệu mà có quyền truy cập đến cơ sở dữ liệu tại bất cứ vị trí nào.
- Các thao tác để **lấy hoặc cập nhật một dữ liệu từ xa** được **tự động thực hiện** bởi hệ thống tại điểm đưa ra yêu cầu.
- Tính trong suốt về vị trí rất hữu ích, nó cho phép người sử dụng **bỏ qua các bản sao dữ liệu** đã tồn tại ở mỗi vị trí.
- Có thể **di chuyển** một bản sao dữ liệu từ một vị trí này đến một vị trí khác và cho phép tạo các bản sao mới mà không ảnh hưởng đến các ứng dụng.

Tính trong suốt phân tán

Trong suốt về vị trí (*Location transparency*):

- *Ví dụ*: với quan hệ tổng thể R và các phân đoạn như đã nói ở trên. Nhưng giả sử rằng DBMS cung cấp trong suốt về vị trí nhưng không cung cấp trong suốt về phân đoạn.

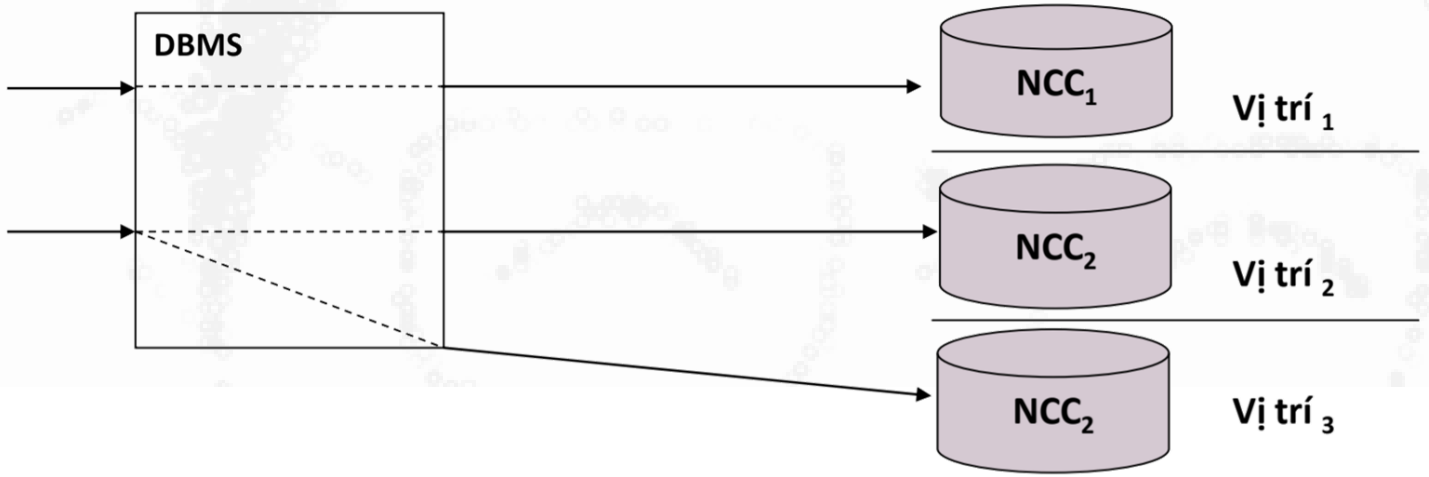
  Xét câu truy vấn tìm người có ID = "ID1".

```
select *
from NCC1
where ID = "ID1"
if NOT #FOUND then
    select *
    from NCC2
    where ID = "ID1"
```

Tính trong suốt phân tán

Trong suốt về vị trí (*Location transparency*):

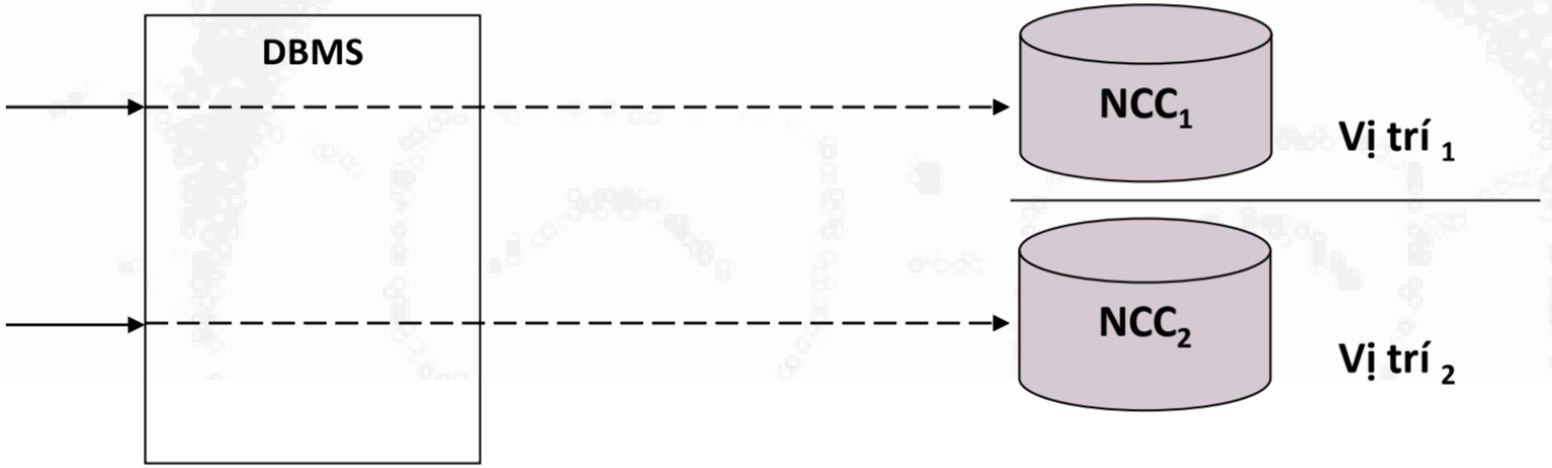
- Đầu tiên hệ thống sẽ thực hiện tìm kiếm ở phân đoạn NCC1 và nếu DBMS trả về biến điều khiển #FOUND thì một câu lệnh truy vấn tương tự được thực hiện trên phân đoạn NCC2 ...
- Ở đây quan hệ NCC2 được sao làm hai bản trên hai vị trí 2 và vị trí 3, ta chỉ cần tìm thông tin trên quan hệ NCC2 mà không cần quan tâm nó ở vị trí nào.



Tính trong suốt phân tán

Trong suốt ánh xạ địa phương (*Local mapping transparency*):

- Là một **đặc tính quan trọng** trong một hệ thống DBMS **không đồng nhất**.
- 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**.



Tính trong suốt phân tán

Sự trong suốt phân tán theo ba mức đối với **hai ứng dụng**:

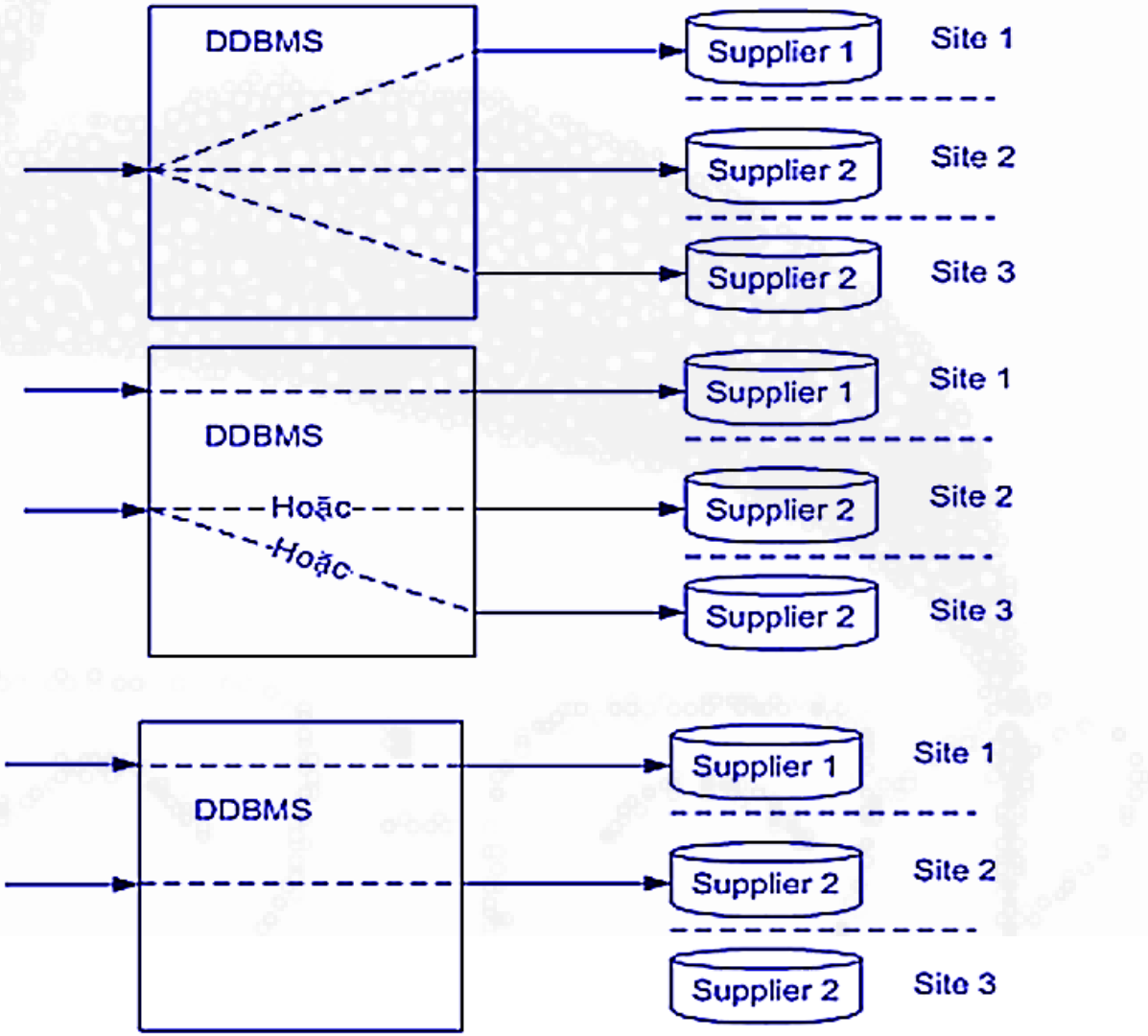
- ❑ Ứng dụng **chỉ đọc**
- ❑ Ứng dụng **cập nhật dữ liệu**

Tính trong suốt phân tán cho ứng dụng chỉ đọc

Mức 1: Trong suốt phân đoạn

Mức 2: Trong suốt về vị trí

Mức 3: Trong suốt ánh xạ địa phương



Example DDB

- Lược đồ toàn cục:

  emp (empnum, name, sal, tax, mgrnum, deptnum)
  dept (deptnum, name, area, mrgnum)
  supplier (snum, name, city)
  suppy (snum, pnum, deptnum, quan)

- Lược đồ phân mảnh:

$$\boldsymbol{emp_1} = \boldsymbol{\sigma}_{deptnum \leq 10} \boldsymbol{\pi}_{empnum,name,mgrnum,deptnum} \boldsymbol{emp}$$

$$\boldsymbol{emp_2} = \boldsymbol{\sigma}_{10 < deptnum \leq 20} \boldsymbol{\pi}_{empnum,name,mgrnum,deptnum} \boldsymbol{emp}$$

$$\boldsymbol{emp_3} = \boldsymbol{\sigma}_{deptnum \geq 20} \boldsymbol{\pi}_{empnum,name,mgrnum,deptnum} \boldsymbol{emp}$$

$$\boldsymbol{emp_4} = \boldsymbol{\pi}_{empnum,name,sal,tax} \boldsymbol{emp}$$

Example DDB

- Lược đồ phân mảnh (tt):

$$\boldsymbol{dept_1} = \boldsymbol{\sigma}_{deptnum \leq 10}\boldsymbol{dept}$$
$$\boldsymbol{dept_2} = \boldsymbol{\sigma}_{10 < deptnum \leq 20}\boldsymbol{dept}$$
$$\boldsymbol{dept_3} = \boldsymbol{\sigma}_{deptnum > 20}\boldsymbol{dept}$$

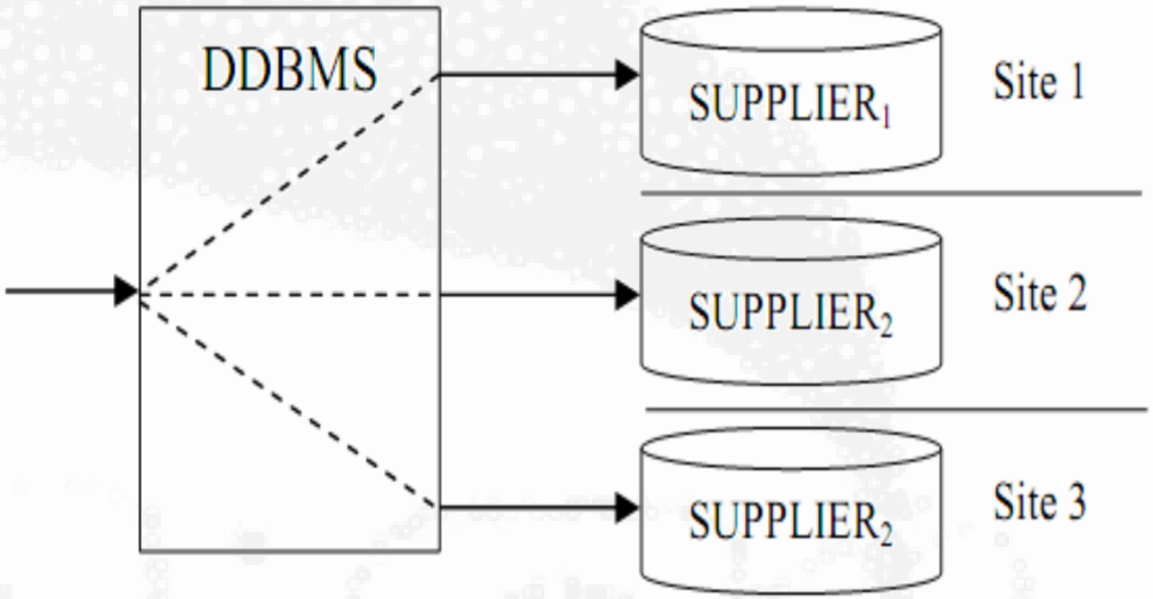
$$\boldsymbol{supplier_1} = \boldsymbol{\sigma}_{city = 'SF'}\boldsymbol{supplier}$$
$$\boldsymbol{supplier_2} = \boldsymbol{\sigma}_{city = 'LA'}\boldsymbol{supplier}$$

$$\boldsymbol{supply_1} = \boldsymbol{supply} \ltimes_{snum = snum} \boldsymbol{supplier_1}$$
$$\boldsymbol{supply_2} = \boldsymbol{supply} \ltimes_{snum = snum} \boldsymbol{supplier_2}$$

Tính trong suốt phân tán cho ứng dụng chỉ đọc

***Mức 1*: Sự trong suốt phân mảnh**

- *Nhận xét*: Câu truy vấn mức 1 **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 cho ứng dụng chỉ đọc

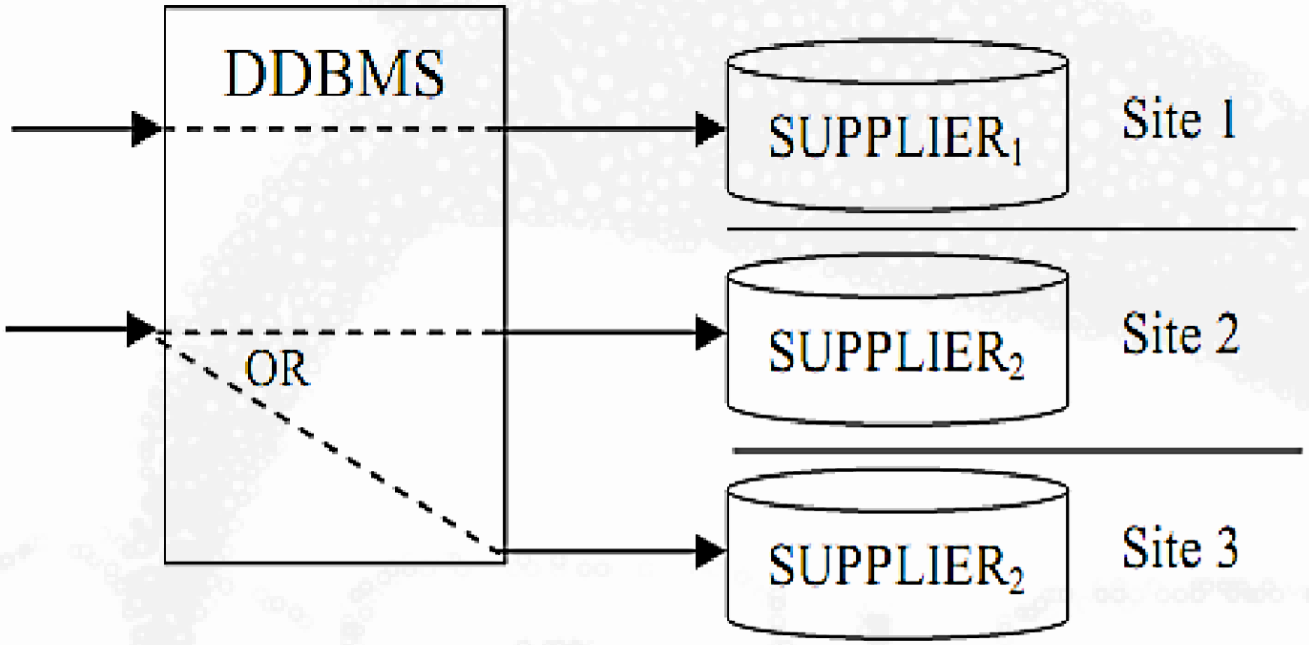
Ví dụ 1: Cho biết **tên của nhà cung cấp** có mã được nhập từ thiết bị đầu cuối.

- 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 cho ứng dụng chỉ đọc

Mức 2: Sự trong suốt vị trí



- *Nhận xét*: Người sử dụng phải **cung cấp 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 cho ứng dụng chỉ đọc

- 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');
```

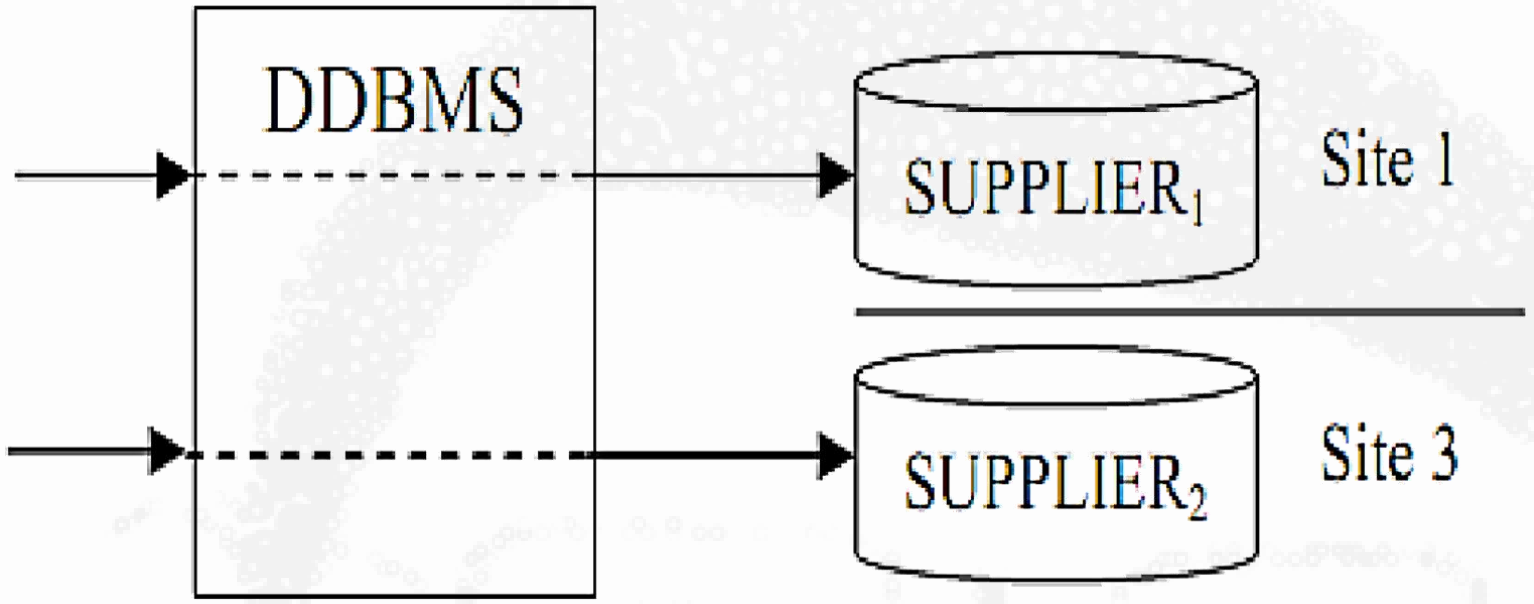
Tính trong suốt phân tán cho ứng dụng chỉ đọc

- Trường hợp dữ liệu nhập có liên quan đến vị từ định tính của 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 cho ứng dụng chỉ đọc

Mức 3: Sự trong suốt ánh xạ cục bộ



- *Nhận xét*: Tại mức trong suốt này, người sử dụng phải **cung cấp các phân mảnh** và **vị trí cấp phát** của chúng.

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- 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 3
        where snum = $snum;
if #FOUND then write (terminal, $name)
else write (terminal, 'Not Found');
```

Tính trong suốt phân tán 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**.

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- 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');
```

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- 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
    and supply2.pnum = $pnum;
if #FOUND then write (terminal, $name)
else write (terminal, 'Not Found');
```

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- 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:

supplier1: tại site **1**

supplier2: tại site **2**

supply1: tại site **3**

supply2: tại site **4**

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- Mức 3 - Trong suốt ánh xạ cục bộ

```
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
    if #FOUND then write (terminal, $name)
    else write (terminal, 'Not Found');
end
```

Tính trong suốt phân tán cho ứng dụng chỉ đọc

- Mức 3 - Trong suốt ánh xạ cục bộ

```
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
        if #FOUND then write (terminal, $name)
        else write (terminal, 'Not Found');
    end
    else write (terminal, 'Not Found');
end
```

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Cập nhật dữ liệu (thêm, sửa, xóa) phải **bảo đảm các ràng buộc toàn vẹn** về khóa chính, khóa ngoại, phụ thuộc hàm, ràng buộc nghiệp vụ ...
- Quy tắc **read-one write-all**.
- Quy tắc **owner – member**.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Quy tắc **read-one write-all**
  - Việc **lấy dữ liệu** có thể được **thực hiện trên bất kỳ một bản sao nào**.
  - Việc **cập nhật mục dữ liệu** phải được **thực hiện trên tất cả các bản sao** của nó.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Quy tắc **owner** - **member** dùng để **di chuyển dữ liệu**
  - **Thêm** vào quan hệ **chủ trước**, thêm vào quan hệ **bộ phận sau**.
  - **Xóa** trong quan hệ **bộ phận trước**, xóa trong quan hệ **chủ sau**.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Sửa dữ liệu** trong cơ sở dữ liệu phân tán
  - ✓ Nếu mục dữ liệu bị sửa **không có trong vị từ định tính**, việc sửa dữ liệu **giống như trong cơ sở dữ liệu tập trung**.
  - ✓ Nếu mục dữ liệu bị sửa **có trong vị từ định tính** và **giá trị của vị từ định tính không bị thay đổi** khi thay thế dữ liệu cũ và dữ liệu mới thì việc sửa dữ liệu **giống như trong cơ sở dữ liệu tập trung**.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Sửa dữ liệu** trong cơ sở dữ liệu phân tán
  - ✓ Nếu mục dữ liệu bị sửa **có trong vị từ định tính** và **giá trị của vị từ định tính bị thay đổi** khi thay thế dữ liệu cũ và dữ liệu mới thì việc **sửa dữ liệu phải di chuyển dữ liệu giữa các mảnh**.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Ví dụ:** Sửa dữ liệu của nhân viên có **mã 100**, từ mã **phòng 3** thành mã **phòng 15**.

  Các mảnh:

  - **emp_1** được đặt tại nơi **1** và **5**.
  - **emp_2** được đặt tại nơi **2** và **6**.
  - **emp_3** được đặt tại nơi **3** và **7**.
  - **emp_4** được đặt tại nơi **4** và **8**.

Example DDB

- Lược đồ toàn cục:

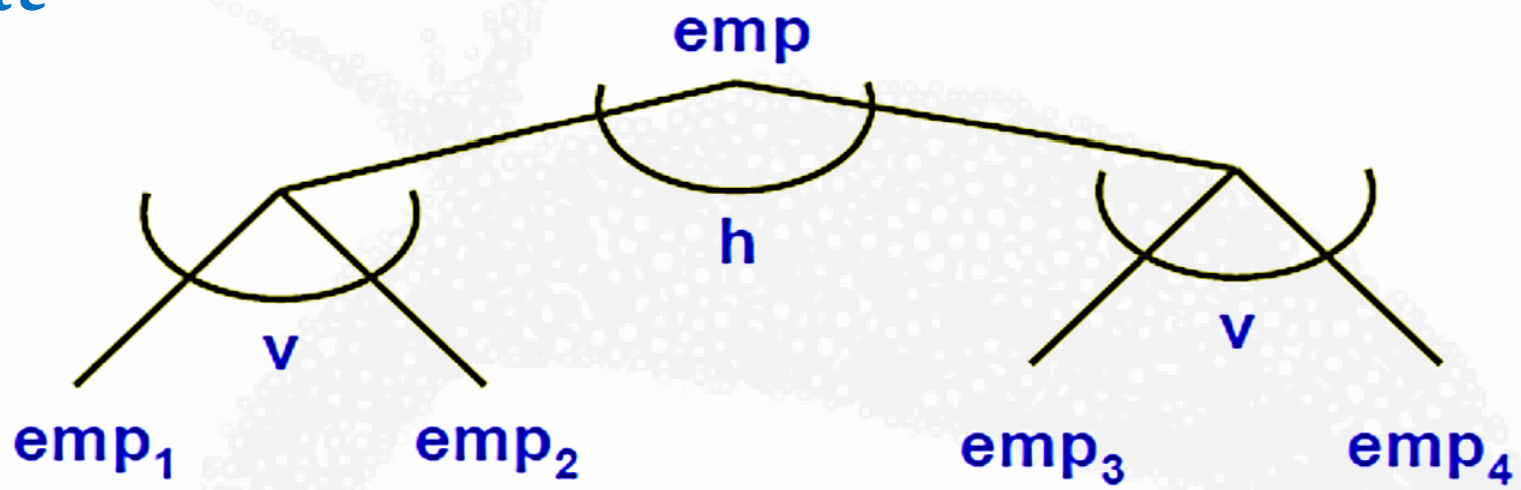
  emp (empnum, name, sal, tax, mgrnum, deptnum)

  dept (deptnum, name, area, mrgnum)

  supplier (snum, name, city)

  suppy (snum, pnum, deptnum, quan)

Tính trong suốt phân tán dùng cho ứng dụng cập nhật



$emp_1 = \Pi_{empnum, name, sal, tax}\ \sigma_{deptnum <= 10}\ emp$

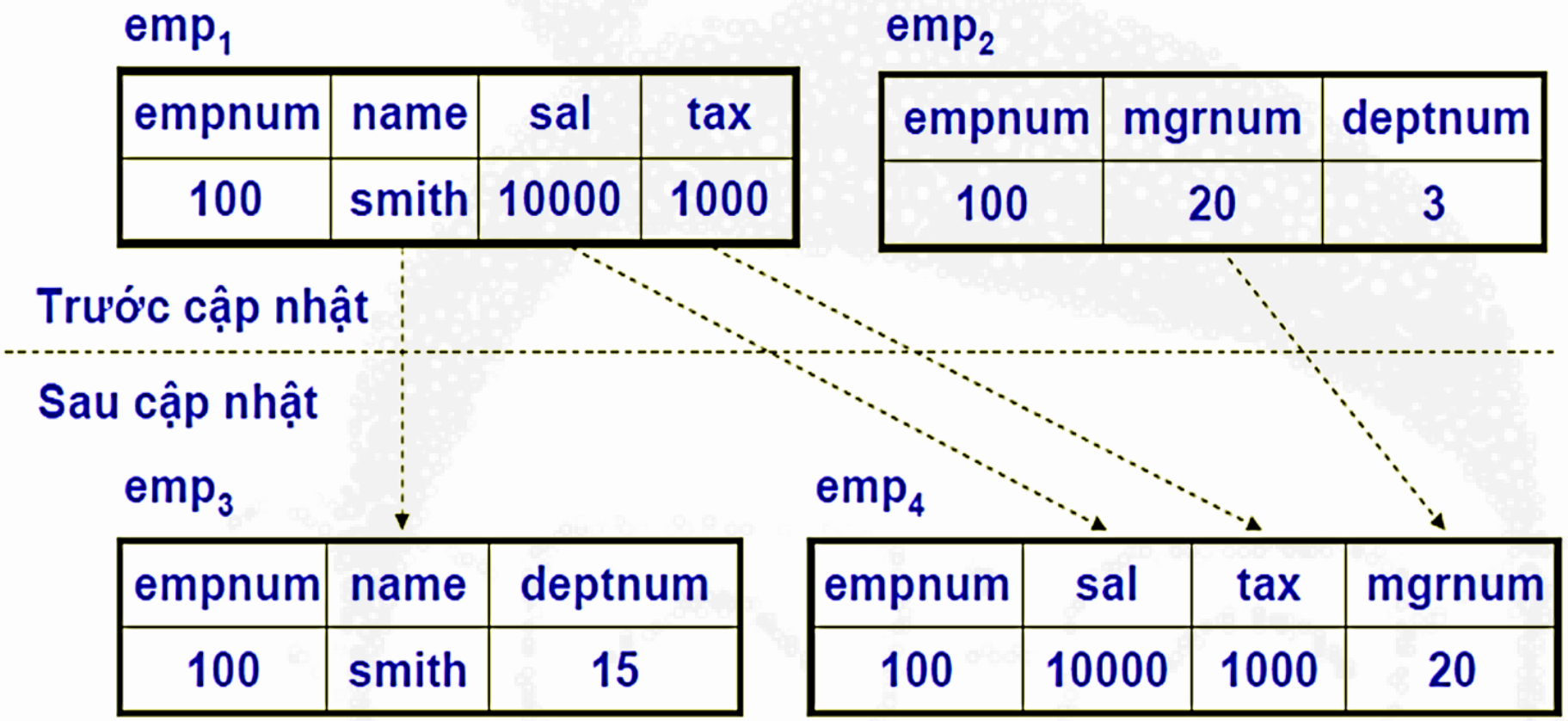
$emp_2 = \Pi_{empnum, mgrnum, deptnum}\ \sigma_{deptnum <= 10}\ emp$

$emp_3 = \Pi_{empnum, name, deptnum}\ \sigma_{deptnum > 10}\ emp$

$emp_4 = \Pi_{empnum, sal, tax, mgrnum}\ \sigma_{deptnum > 10}\ emp$

Cây phân mảnh của quan hệ EMP

Tính trong suốt phân tán dùng cho ứng dụng cập nhật



Ứng dụng cập nhật

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Mức 1 - Trong suốt phân mảnh

```
update emp
set deptnum = 15
where empnum = 100;
```

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Mức 2 - Trong suốt vị trí

```
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;
```

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Mức 2 - Trong suốt vị trí

```
insert into emp3 (empnum, name, deptnum)
values (100, $name, 15);
insert into emp4 (empnum, sal, tax, mgrnum)
values (100, $sal, $tax, $mgrnum);
delete from emp1 where empnum = 100;
delete from emp2 where empnum = 100;
end
```

Tính trong suốt phân tán dùng cho 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.
  - Giả sử các phân mảnh của quan hệ EMP được cấp phát như sau:

    emp_1 được đặt tại nơi **1** và **5**.

    emp_2 được đặt tại nơi **2** và **6**.

    emp_3 được đặt tại nơi **3** và **7**.

    emp_4 được đặt tại nơi **4** và **8**.

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- Mức 3: Trong suốt ánh xạ cục bộ

```
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);
```

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

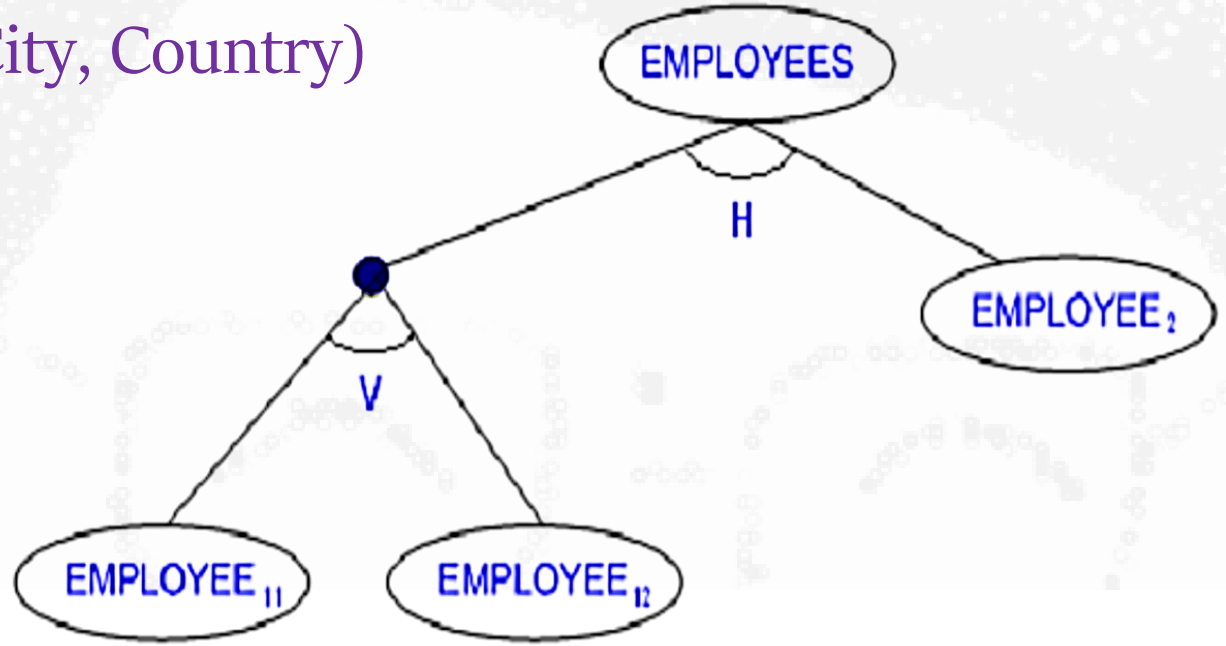
- Mức 3: Trong suốt ánh xạ cục bộ

```
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 from emp1 at site 1 where empnum = 100;
delete from emp1 at site 5 where empnum = 100;
delete from emp2 at site 2 where empnum = 100;
delete from emp2 at site 6 where empnum = 100;
```

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Ví dụ 2:** Cho cây phân mảnh của quan hệ:

EMPLOYEES (EmployeeID, LastName, FirstName, BirthDate, HireDate, Address, City, Country)



Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Ví dụ 2:**

Trong đó:

- *Site 2:* EMPLOYEE2 = $\sigma_{\text{COUNTRY <> 'USA'}}$ (**EMPLOYEES**)
- *Site 1:* EMPLOYEE11 = $\pi_{\text{EmployeeID, LastName, FirstName, BirthDate}}$ ($\sigma_{\text{Country = 'USA'}}$ (**EMPLOYEES**))
- *Site 3:* EMPLOYEE12 = $\pi_{\text{EmployeeID, HireDate, Address, City, Country}}$ ($\sigma_{\text{Country = 'USA'}}$ (**EMPLOYEES**))

Tính trong suốt phân tán dùng cho ứng dụng cập nhật

- **Ví dụ 2:**

Ta muốn cập nhật một bộ có giá trị EmployeeID = 8 như sau:

- Bộ này nằm ở hai phân mảnh là EMPLOYEE11, EMPLOYEE12.
- Ta cập nhật country của bộ này từ 'USA' thành 'UK'.

EmployeeID	LastName	FirstName	BirthDate
8	Callahan	Laura	01/09/1958

EmployeeID	HireDate	Address	City	Country
8	03/05/1994	4726 – 11th Ave. N.E.	Seattle	USA