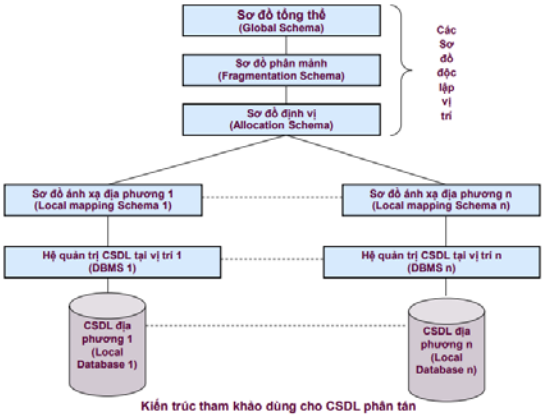
Cơ sở dữ liệu phân tán

Phạm Minh Khan
pmkhan@hcmunre.edu.vn

Chương 2: Sự trong suốt trong cơ sở dữ liệu phân tán

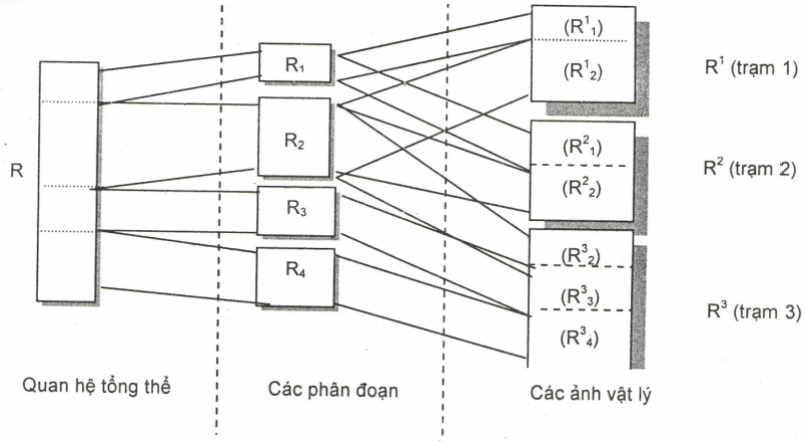
1. Khái niệm về phân tán dữ liệu, các kiểu phân mảnh, quy tắc phân mảnh.
2. Sự trong suốt phân tán trong việc truy xuất dữ liệu

Kiến trúc tham khảo dùng cho CSDL phân tán



Kiến trúc tham khảo dùng cho CSDL phân tán

Kiến trúc tham khảo dùng cho CSDL phân tán



Các đoạn và các ảnh vật lý của một quan hệ tổng thể

Khái niệm về phân tán dữ liệu, các kiểu phân mảnh, quy tắc phân mảnh.

Các câu hỏi sau đây sẽ bao quát toàn bộ vấn đề:

- Tại sao cần phải phân mảnh?
- Làm thế nào để thực hiện phân mảnh?
- Phân mảnh nên thực hiện đến mức độ nào?
- Có cách gì kiểm tra tính đúng đắn của phân mảnh hay không?
- Chúng ta sẽ cấp phát như thế nào?
- Những thông tin nào cần thiết cho việc cấp phát?

Các lý do phân mảnh

Các lý do cần phân mảnh:

- Ứng dụng không cần truy xuất đến toàn bộ quan hệ mà chỉ tập con của quan hệ
- Một quan hệ do được chia nhỏ ra ở từng phân mảnh nên sẽ có phép xử lý nhiều giao tác đồng thời.
- Đồng thời việc phân mảnh các quan hệ sẽ cho phép thực hiện song song một truy vấn bằng cách chia nhỏ nó thành một tập các câu vấn tin con hoạt tác trên từng mảnh => làm tăng mức độ hoạt động song hành và tăng lưu lượng hoạt động của hệ thống.

Khuyết điểm:

- Thời gian truy vấn dữ liệu từ hai phân mảnh rồi nối hoặc hợp chúng lại thì chi phí rất cao
- Tính toàn vẹn dữ liệu

Các kiểu phân mảnh

Các kiểu phân mảnh

- Phân mảnh ngang
- Phân mảnh dọc
- Phân mảnh hỗn hợp

Mức độ phân mảnh

- Bao nhiêu phân mảnh là hợp lý ?
- Những ảnh hưởng khi phân mảnh
  - Nếu phân tán nhiều mảnh quá thì ưu điểm (dữ liệu ít sẽ truy vấn nhanh), khuyết điểm (phải quản lý các phân mảnh đó)
  - Nếu số mảnh ít thì dữ liệu nhiều, nhưng hiệu suất truy vấn sẽ lâu hơn.

Các quy tắc phân mảnh

Trong quá trình phân mảnh chúng ta phải tuân thủ **ba qui tắc** để bảo đảm cơ sở dữ liệu sẽ không thay đổi về mặt ngữ nghĩa.

- Tính đầy đủ (completeness): Nếu một **quan hệ R** được phân rã thành các mảnh $R_1, R_2, \ldots, R_n$ thì mỗi **mục dữ liệu** trong R cũng có thể có trong một hay nhiều mảnh R_i . Đặc tính này nói lên sự **phân mảnh mà không mất thông tin**.
- Tính tái thiết được (reconstruction): Nếu một **quan hệ R** được phân rã thành các mảnh $R_1, R_2, \ldots, R_n$ thì cần định nghĩa một phép toán quan hệ ∇ sao cho:

$$R = \nabla R_i \; i \; [1,n]$$

Phép toán ∇ thay đổi tùy theo loại phân mảnh. Khả năng tái thiết một quan hệ từ các mảnh của nó bảo đảm rằng các ràng buộc được định nghĩa trên dữ liệu dưới dạng phụ thuộc hàm sẽ được bảo toàn.

Các quy tắc phân mảnh

- Tính tách biệt (disjointness): Nếu một quan hệ R được phân rã thành các mảnh $R_1, R_2, \ldots, R_n$ và mục dữ liệu d_i nằm trong mảnh R_j , thì nó sẽ không nằm trong một mảnh R_k khác ($k \neq j$).

Tiêu chuẩn này đảm bảo các mảnh ngang sẽ tách biệt. Nếu quan hệ được phân rã dọc, các thuộc tính khoá chính phải được lặp lại trong mỗi mảnh.

Các kiểu cấp phát

	Nhân bản hoàn toàn	Nhân bản 1 phần	Phân hoạch
Xử lý vấn tin	Dễ	Khó ⟷	
Quản lý từ điển dữ liệu	Dễ	Khó ⟷	
Điều khiển đồng thời	Khó	Vừa phải	Dễ
Độ khả tín	Rất cao	Cao	Thấp
Tính thực tế	Thực tế	Thực tế	Thực tế

Các phương pháp phân mảnh

❖ Phân mảnh ngang

Phân mảnh ngang chia một quan hệ theo các bộ (dòng, mẫu tin). Vì vậy mỗi mảnh là một tập con của quan hệ. Có hai loại phân mảnh ngang: phân mảnh ngang nguyên thủy và phân mảnh ngang dẫn xuất.

- **Phân mảnh ngang nguyên thủy** (Primary Horizontal Fragmentation) là sự phân mảnh một quan hệ ***dựa trên một vị từ*** được định nghĩa trên một quan hệ.
- **Phân mảnh ngang dẫn xuất** (Derived Horizontal Fragmentation) là phân rã một quan hệ ***dựa vào các vị từ được định nghĩa trên một quan hệ khác***.

Cho quan hệ R, các mảnh ngang Ri là: $\mathbf{Ri} = \sigma_{Fi}(\mathbf{R})$

Trong đó Fi là công thức chọn để có được mảnh Ri .

Các phương pháp phân mảnh

Ví dụ: Cho lược đồ quan hệ toàn cục :

SUPPLIER (SNUM,NAME,CITY)

Chúng ta có thể có hai phân mảnh ngang sau:

SUPPLIER1 = $\sigma_{\text{CITY= "HCM"}}$ (SUPPLIER)

SUPPLIER2 = $\sigma_{\text{CITY= "HANOI"}}$ (SUPPLIER)

Phân mảnh này thỏa:

- Tính đầy đủ: "HCM" và "HANOI" đều xuất hiện trên 2 phân mảnh
- Tính tái thiết được kiểm tra dễ dàng vì có thể tái thiết lại quan hệ toàn cục SUPPLIER bằng phép toán hội: SUPPLIER = SUPPLIER1 **U** SUPPLIER2
- Điều kiện tách biệt cũng được kiểm tra một cách rõ ràng

Các phương pháp phân mảnh

Phân mảnh ngang dẫn xuất

Trong một số trường hợp sự **phân mảnh ngang được dẫn ra từ một phân mảnh ngang của một quan hệ khác**.

Ví dụ: Một quan hệ toàn cục

SUPPLY (SNUM, PNUM, DEPTNUM, QUAN)

Sự phân mảnh dẫn xuất của SUPPLY có thể được định nghĩa như sau:

SUPPLY1 = SUPPLY SJ SUPPLIER1

SUPPLY2 = SUPPLY SJ SUPPLIER2

Với SJ là phép toán nửa kết (Semi Join)

- Tính tái thiết quan hệ toàn cục SUPPLY có thể được thể hiện qua phép hội.
- Rõ ràng với phép kết như vậy thì một bộ trong quan hệ SUPPLY chỉ có thể thuộc 1 trong 2 mảnh SUPPLY1 hoặc SUPPLY2 , do đó, nó thỏa tính đầy đủ và tách biệt .

Các phương pháp phân mảnh

❖ **Phân mảnh dọc**

Sự phân mảnh dọc của một quan hệ toàn cục là việc chia các thuộc tính vào hai nhóm; các mảnh nhận được từ phép chiếu quan hệ toàn cục trên mỗi nhóm. 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.

Các phương pháp phân mảnh

❖ **Phân mảnh dọc**

Ví dụ: Xét quan hệ toàn cục

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

Một sự phân mảnh dọc của quan hệ này được định nghĩa:

$EMP_1 = \pi_{EMPNUM,\ NAME,\ MGRNUM,\ DETPNUM}\ (EMP)$

$EMP_2 = \pi_{EMPNUM,\ SAL,\ TAX}\ (EMP)$

Phân mảnh này phản ánh lương và thuế của các nhân viên được quản lý riêng. Việc tái thiết lại quan hệ EMP có thể nhận được từ :

$EMP = EMP_1$ JNN EMP_2

(với JNN là phép kết nối tự nhiên hai quan hệ)

=> Phân mảnh cũng thỏa tính đầy đủ và tính tách biệt.

Các phương pháp phân mảnh

❖ Phân mảnh hỗn hợp

Ví dụ: Xét quan hệ toàn cục:

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

Dưới đây là một sự phân mảnh hỗn hợp bằng cách áp dụng sự phân mảnh dọc rồi sau đó áp dụng phân mảnh ngang trên DEPTNUM:

$EMP_1 = \sigma_{deptnum<=10} (\pi_{empnum, name, mgrnum, deptnum} (EMP))$

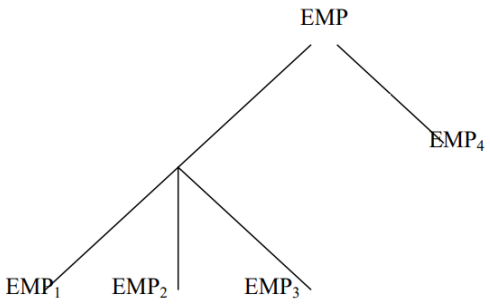
$EMP_2 = \sigma_{10<deptnum <=20}(\pi_{empnum, name, mgrnum, deptnum}(EMP))$

$EMP_3 = \sigma_{deptnum>20} (\pi_{empnum, name, mgrnum, deptnum} (EMP))$

$EMP_4 = \pi_{empnum, name, sal, tax} (EMP)$

Các phương pháp phân mảnh

❖ Phân mảnh hỗn hợp



Cây phân mảnh của quan hệ EMP

Việc tái thiết lại quan hệ EMP được định nghĩa bởi biểu thức:

$EMP = U(EMP_1, EMP_2 , EMP_3)$ JNN $\pi_{empnum, sal, tax}$ (EMP4)

Sự trong suốt phân tán

Để hiểu được sự trong suốt phân tán

Xét câu lệnh phân tán được "trong suốt phân tán":

Định nghĩa: Trong suốt nghĩa là 1 SP khi ta cho thực thi ở 1 Server phân mảnh bất kỳ thì vẫn thực thi được ở tất cả các Server phân mảnh còn lại mà ta không cần chỉ lại đường dẫn đến SP cần truy xuất (**người dùng không cảm nhận được là SP đang chạy trên hệ thống phân tán**)

- Bằng cách nào để ta thực hiện điều đó: có nghĩa khi ta chạy trên bất kỳ 1 phân mảnh nào thì không cần chỉ rõ nơi chưa SP đó ➔ **Nhân bản tập lệnh SP tới các server mà mình thấy cần thiết.**
- Hỗ trợ đơn giản để viết SP trong suốt:
  - Tên DB ở các subcriber phải giống
  - Tên link server phải giống nhau

Sự trong suốt phân tán của ứng dụng chỉ đọc

Để hiểu sự trong suốt phân tán của ứng dụng chỉ đọc thì chúng ta sẽ xem xét các ví dụ sau

Ví dụ:

- Các biến có kiểu chuỗi ký tự. (Declare @manv nvarchar(50))
- Nhập xuất được thực hiện qua các thủ tục chuẩn:

  Read(file name, variable)

  Set @bien = giá trị hoặc gởi giá trị cho tham số của SP

  Write (file name, variable)

  Select @bien

Sự trong suốt phân tán của ứng dụng chỉ đọc

Ví dụ: Xét câu truy vấn : tìm tên của người cung cấp khi biết mã số người cung cấp

Set @Name = (Select name from Supplier Where Snum = @SNum)

Set @City = (Select City From Supplier Where Snum = @SNum)

➔ Ghi ngắn gọn lại

Set @Name = Name, @City = City From Supplier Where Snum = @Snum

Trong câu lệnh truy vấn này thì @Name, @City là biến xuất, còn @Snum là biến nhập

Sự trong suốt phân tán của ứng dụng chỉ đọc

Các biến toàn cục của hệ quản trị cở dữ liệu: thực chất là các hàm có sẵn trong Sql Server: **không thể gán giá trị cho biến, biến toàn cục không có kiểu, tên biến bắt đầu bằng @@**

Các biến hệ thống của sẽ bắt đầu bằng dấu @@

Ví dụ: *@@ROWCOUNT*: kết quả truy vấn gần nhất hoặc trả về số dòng bị ảnh hưởng bởi lệnh thực thi gần nhất

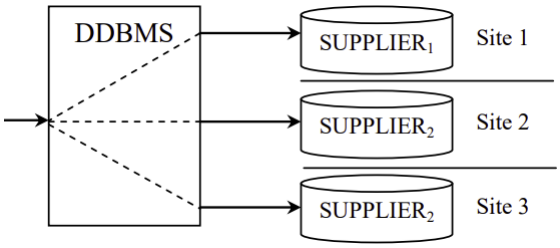
```
Update Employees set LastName = 'Trần'
Where LastName ='Nguyễn'
If(@@rowcount=0)
begin
    print 'Không dòng nào được cập nhật'
    return
end
```

Các mức độ trong suốt

Xét trên một lược đồ quan hệ phổ quát:

➢**Mức 1:** Sự trong suốt phân tán

```
Read (terminal, @SNUM)
        Select NAME into @NAME
        From SUPPLIER
        Where SNUM = @SNUM
Write(terminal, @NAME)
```



Sự trong suốt phân tán

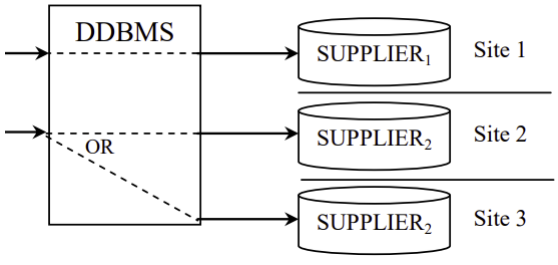
➔Khi đó người sử dụng không hề có cảm giác là đang thao tác trên một câu truy vấn phân tán.

Các mức độ trong suốt

Xét trên một lược đồ quan hệ phổ quát:

➢**Mức 2:** Sự 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
Write(terminal, @NAME)
```



Sự trong suốt vị trí

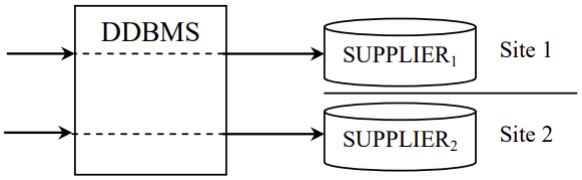
➔ Người sử dụng phải cung cấp tên các phân mảnh cụ thể cho câu truy vấn nhưng không cần chỉ ra vị trí của các phân mảnh

Các mức độ trong suốt

Xét trên một lược đồ quan hệ phổ quát:

➢**Mức 3:** Sự 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
Write(terminal, @NAME)
```



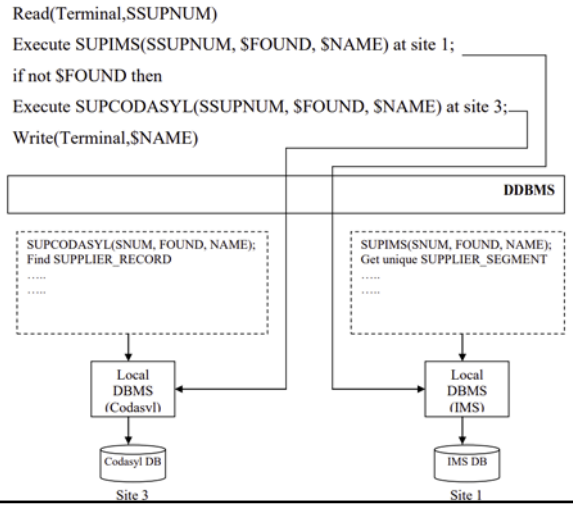
Sự trong suốt ánh xạ cục bộ

➔ Tại mức trong suốt này người sử dụng phải cung cấp tên các phân mảnh và vị trí cấp phát của chúng.

Các mức độ trong suốt

Xét trên một lược đồ quan hệ phổ quát:

➢**Mức 4:** Không trong suốt



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

Các mức độ trong suốt

Xét trên hai lược đồ quan hệ phổ quát:

Chúng ta hãy xét một ví dụ phức tạp hơn. Giả sử cần tìm tên những nhà cung cấp mặt hàng có mã số (mặt hàng) cho trước.

➢**Mức 1:** Trong suốt phân tán

```
Read(Terminal, @PNUM)
   Select @NAME = NAME
   From SUPPLIER, SUPPLY
   Where SUPPLIER.SNUM = SUPPLY.SNUM and SUPPLY.PNUM = @PNUM
Write(Terminal, @NAME)
```

=>Bản thân câu lệnh này là lệnh xuất rồi thì không cần Write ở đây

Các mức độ trong suốt

➢**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
Write(Terminal, $NAME)
```

Các mức độ trong suốt

➢**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

```
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
            send $SNUM  from site 4 to site 2
            Select NAME into $NAME from SUPPLIER2 at site 2
            where SUPPLIER2.SNUM = $SNUM
        end;
Write(Terminal, $NAME)
```

Sự trong suốt phân tán đối với các ứng dụng cập nhật

Những vấn đề khi cập nhật trong ứng dụng phân tán:

- Đối với những ứng dụng cập nhật khi ta lập trình mà câu lệnh của ta tác động lên nhiều mẫu tin của 1 bảng hay nhiều bảng thì ta phải chú ý đến vấn đề là **giao tác cập nhật.**
- **Các mức trong suốt vẫn tương tự như ứng dụng chỉ đọc** nhưng lưu ý khi ta cập nhật dữ liệu ở trên phân mảnh thì dữ liệu đó có khả năng là **nhân bản hoàn toàn, nhân bản 1 phần** khi ta cập nhật dữ liệu 1 quan hệ trên phân mảnh đó thì ta cũng phải cập nhật dữ liệu đó ở site chủ và trên những phân mảnh còn lại =>Vấn đề này khá phức tạp.
- Khi viết SP, cập nhật số liệu thì vấn đề di chuyển dữ liệu sau khi cập nhật

Các nguyên tố truy xuất cơ sở dữ liệu phân tán

- Sau khi đã có dữ liệu đổ dữ liệu vào một quan hệ tạm thì chúng ta sẽ qui ước dùng tham số có tiếp vị ngữ "REL"

Ví dụ: Select EMPNUM, NAME INTO #EMP_REL($EMPNUM, $NAME) from EMP

Tài liệu tham khảo

- Tài liệu giảng dạy cơ sở dữ liệu phân tán của PIIT
- M.TAMER OZSU and PATRICK VALDURIEZ, Principles of Distributed Database Systems, Springer, 2020
- Cơ sở dữ liệu phân tán, PGS.TS Nguyễn Mậu Hân

Thank you for listening