HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CƠ SỞ DỮ LIỆU PHÂN TÁN

THIẾT KẾ
CƠ SỞ DỮ LIỆU PHÂN TÁN
Phân mảnh dọc

Ts. Phan Thị Hà

Phân mảnh dọc

Định nghĩa

Phân mảnh dọc quan hệ R sinh ra các mảnh R_1, R_2, ..., R_r, sao cho mỗi mảnh chứa một tập con các thuộc tính của quan hệ R và khoá của nó.

Mục đích

Phân chia quan hệ R thành các mảnh nhỏ hơn là để cho nhiều ứng dụng có thể thực hiện chỉ trên một mảnh tối ưu, giảm thiểu thời gian thực hiện ứng dụng. Nâng cao hiệu năng xử lý đồng thời.

Tối ưu ?

Một phân mảnh tối ưu là phân mảnh sinh ra một lược đồ phân mảnh cho phép giảm tối đa thời gian thực thi các ứng dụng chạy trên phân mảnh đó

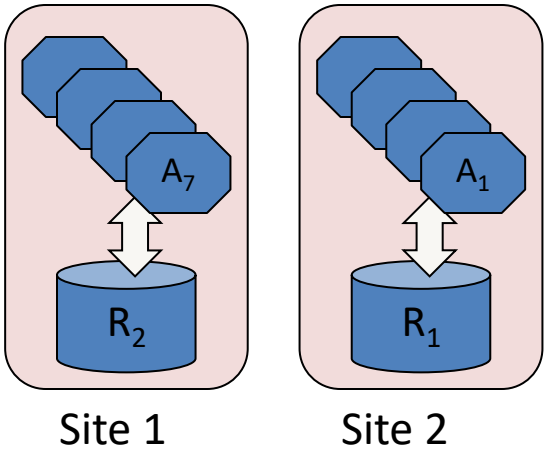
Phân mảnh dọc

Purpose: *Phân mảnh dọc là chúng ta gom những thuộc tính thường được truy xuất chung với nhau vào 1 mảnh.*

VD: Hình bên

Purpose Vd: Xác định các phân mảnh R1, R2, các ứng dụng có thể được thực thi chỉ trên 1 phân mảnh

AdvantageVD: Khi nhiều ứng dụng sử dụng R1và nhiều ứng dụng sử dụng R2 ở các site khác nhau, giảm thiểu thời gian thực hiện ứng dụng. Nâng cao hiệu năng xử lý đồng thời.



- Phân mảnh dọc là chúng ta gom những thuộc tính thường được truy xuất chung với nhau vào 1 mảnh.
  Để tiến hành phân mảnh dọc chúng ta cần 1 số thông tin có liên quan đến ứng dụng (các câu truy vấn) và các quan hệ giữa các thuộc tính, giữa các ứng dụng với các thuộc tính:

- Ma trận sử dụng thuộc tính: biểu diễn mối liên hệ giữa các câu truy vấn và các thuộc tính

- Ma trận lực hút AA: Đo lực hút giữa 2 thuộc tính (Ai,Aj).

-Ma trận lực hút tụ nhóm: đã gom nhóm các thuộc tính lại với nhau (các thuộc tính có số đo lực hút gần bằng nhau thì được xếp kề nhau)

Split Approach phân mảnh dọc

1. Ma trận sử dụng thuộc tính (A)
2. Ma trận lực hút thuộc tính (AA) đc xây dựng từ ma trận A, phụ thuộc vào tần số truy cập của mỗi truy vấn qk vào cặp thuộc tính (Ai,Aj)trên mỗi site và tần số truy cập của qk vào mỗi site
3. Thuật tụ nhóm (BEA) để nhóm các thuộc tính xuất hiện cúng nhau dựa trên ma trận lực hút thuốc tính. Thuật toán này sản xuất ra ma trận lực hút tụ nhóm. (CA)
4. Sử dụng thuật toán Phân mảnh dọc tách các thuộc tính ra thành các nhóm thuộc tính duy nhất

Ma trận giá trị sử dụng thuộc tính

- $R(A_1, A_2, \ldots, A_n)$ quan hệ toàn cục
- $Q=\{q_1, q_2, .., q_m\}$ tập các ứng dụng
- Ma trận giá trị sử dụng thuộc tính định nghĩa như sau:

$$A= (use(q_i , A_j))_{m*n}$$

$$use(q_i,A_j) = \begin{cases} \mathbf{1} & \textbf{Nếu } \mathbf{q_i} \textbf{ tham chiếu đến thuộc tính } \mathbf{A_j} \\ \mathbf{0} & \textbf{Ngược lại} \end{cases}$$

$i=1..m$ và $j=1..n$

$n = |\Omega|$ và $m=|Q|$

Ma trận giá trị sử dụng thuộc tính

$A =$

	A1	A2	.....	An
q1	Use(q1,A1)	Use(q1,A2)		Use(q1,A_n)
q2	Use(q2,A1)	Use(q2,A2)		Use(q2,A_n)
....	.....	.....	...	....
q_m	Use(q_m,A1)	Use(q_m,A2)		Use(q_m,A_n)

Ví dụ ma trận giá trị sử dụng thuộc tính

Quan hệ: PROJ (PNO, PNAME, BUDGET, LOC)

Tập các ứng dụng:

q1: Kinh phí của dự án khi biết mã dự án

SELECT BUDGET FROM PROJ WHERE PNO = Value

q2: Tên và kinh phí của tất các dự án

SELECT PNAME, BUDGET FROM PROJ

q3: Tìm tên các dự án khi biết thành phố

SELECT PNAME FROM PROJ WHERE LOC = Value

q4:Tổng kinh phí của các dự án tại mỗi thành phố

SELECT SUM(BUDGET) FROM PROJ WHERE LOC = Value

Ví dụ ma trận giá trị sử dụng thuộc tính

Ký hiệu: A1= PNO, A2=PNAME, A3=BUDGET, A4=LOC

q1: SELECT A3 FROM PROJ WHERE A1= Value

q2: SELECT A2, A3 FROM PROJ

q3: SELECT A2 FROM PROJ WHERE A4 = Value

q4: SELECT SUM(A3) FROM PROJ WHERE A4= Value

$$\mathbf{A} = \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Trọng số lực hút
(Attribute Affinit Measure)

- R (A_1, A_2,…, A_n) quan hệ toàn cục
- Q={q_1, q_2,.., q_m} tập các ứng dụng
- Các site: S = {S_1, S_2,…,S_t}
- Khi đó AA = (aff(A_i , A_j))$_{n*n}$ Ma trận lực hút
- aff(A_i , A_j): Trọng số lực hút (Ai, Aj) với các ứng dụng trên các Site

$$aff(A_i, A_j) = \sum_{k:[(use(q_k, A_i) \wedge (use(q_k, A_j)]=1} \sum_{\forall S_l} ref_l(q_k) acc_l(q_k)$$

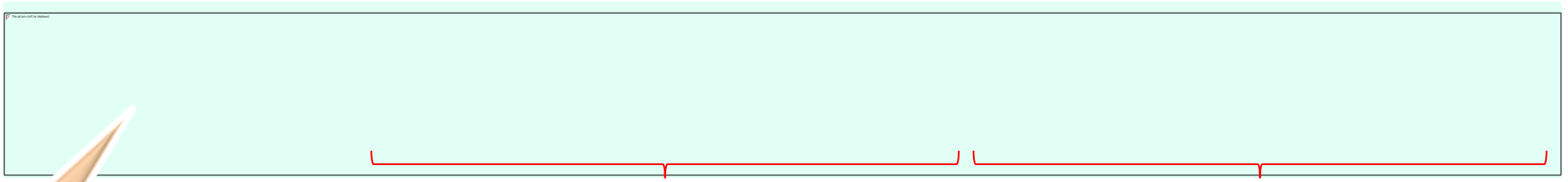
Trong đó:

- $ref_l(q_k)$ là số lượng truy suất trên (A_i,A_j) cho mỗi lần thực hiện của qk trên vị trí S_l
- $acc_l(q_k)$ là tần số truy cập ứng dụng của q_k tại vị trí S_l

Ma trận lực hút AA(Attribute Affinity Matrix)

AA =

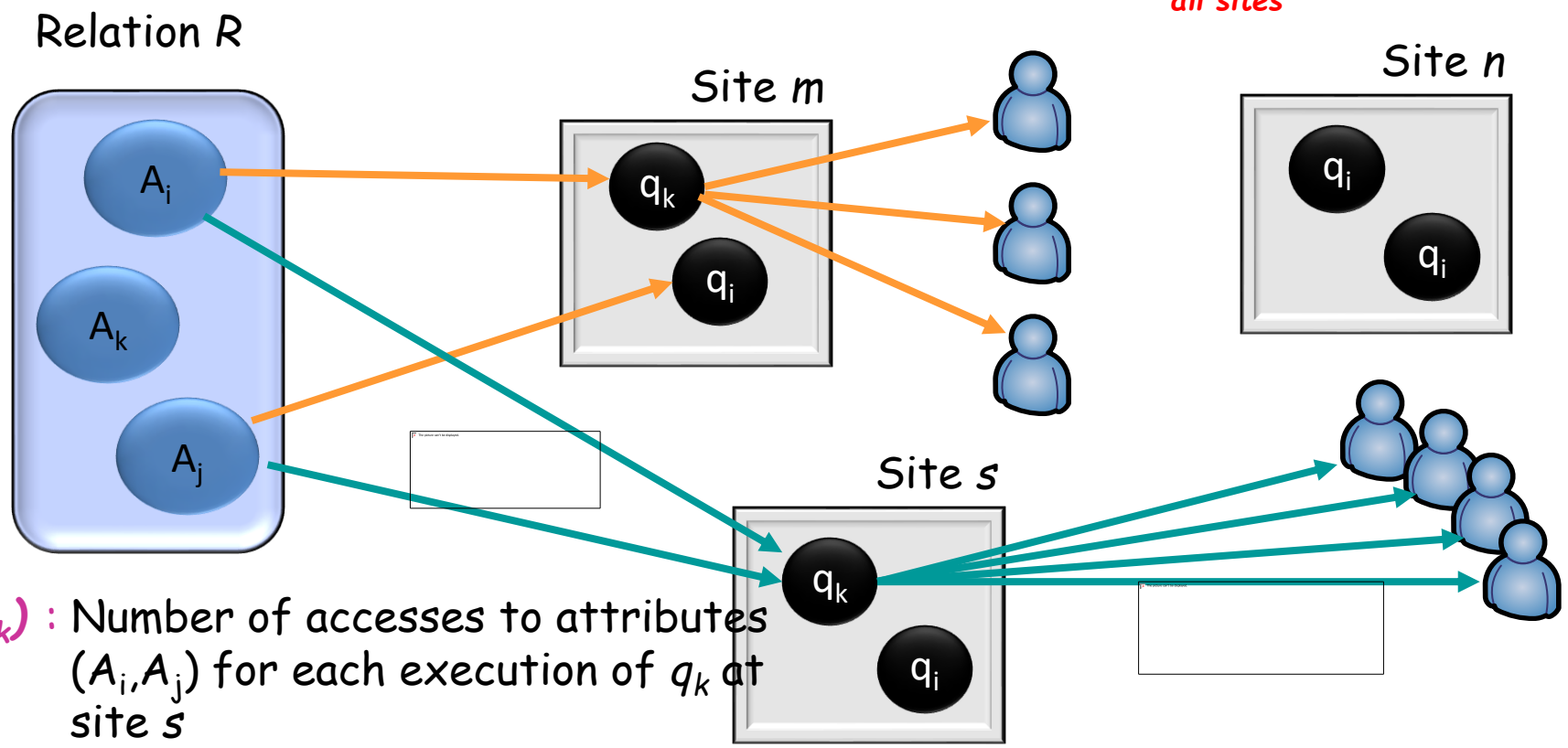
	A1	A2	….	An
A1	aff(A1,A1)	aff(A1,A2)		aff(A1,A_n)
A2	aff(A2,A1)	aff(A2,A2)		aff(A2,A_n)
….	…..	…..	…	….
A_n	aff(A_n,A1)	aff(A_n,A2)		aff(A_n,A_n)



For each query q_k that uses both A_i and A_j

Popularity of such A_i-A_j pair at all sites

Popularity of using A_i and A_j together



$ref_s(q_k)$: Number of accesses to attributes (A_i,A_j) for each execution of q_k at site s

$acc_s(q_k)$: Application access frequency of q_k at site s.

Ví dụ ma trận lực hút AA

- Giả sử $ref_l(q_k) = 1$ cho tất cả q_k và S_l
- Giả sử tần số các ứng dụng trên các Site là:

Site1	**Site2**	**Site3**
$acc_1(q_1)=15$	$acc_2(q_1)=20$	$acc_3(q_1)=10$
$acc_1(q_2)=5$	$acc_2(q_2)=0$	$acc_3(q_2)=0$
$acc_1(q_3)=25$	$acc_2(q_3)=25$	$acc_3(q_3)=25$
$acc_1(q_4)=3$	$acc_2(q_4)=0$	$acc_3(q_4)=0$

Ví dụ ma trận lực hút AA

Site1	Site2	Site3
$acc_1(q_1)=15$	$acc_2(q_1)=20$	$acc_3(q_1)=10$
$acc_1(q_2)=5$	$acc_2(q_2)=0$	$acc_3(q_2)=0$
$acc_1(q_3)=25$	$acc_2(q_3)=25$	$acc_3(q_3)=25$
$acc_1(q_4)=3$	$acc_2(q_4)=0$	$acc_3(q_4)=0$

$\mathbf{A} =$

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

$$aff(A_i, A_j) = \sum_{k:[(use(q_k, A_i) \wedge (use(q_k, A_j)]=1} \; \sum_{\forall S_l} ref_l(q_k) acc_l(q_k)$$

$$aff(A_1, A_3) = \sum_{k=1}^{1} \sum_{l=1}^{3} acc_l(q_k) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

$\mathbf{AA} =$

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

Ví dụ ma trận lực hút AA

AA =

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

Ma trận lực hút tụ nhóm (CA)

- Sử dụng thuật tụ nhóm (BEA) để nhóm các thuộc tính xuất hiện cúng nhau dựa vào ma trận lực hút thuộc tính AA.
- Thuật toán hoán vị các hàng và các cột của ma trận AA, sao cho số đo lực hút chung AM là lớn nhất

$$AM = \sum_i \sum_j (\text{affinity of } A_i \text{ and } A_j \text{ with their neighbors})$$

- Số đo lực hút AM được tính:

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$$

where

$$aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0$$

- Vì ma trận AA đối xứng, nên khi tính AM ta có thể giảm chức năng mục tiêu của AM để

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$$

which can be rewritten as

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} [aff(A_i,A_j)aff(A_i,A_{j-1}) + aff(A_i,A_j)aff(A_i,A_{j+1})]$$

$$= \sum_{j=1}^{n} \left[\sum_{i=1}^{n} aff(A_i,A_j)aff(A_i,A_{j-1}) + \sum_{i=1}^{n} aff(A_i,A_j)aff(A_i,A_{j+1}) \right]$$

Let us define the *bond* between two attributes A_x and A_y as

$$bond(A_x,A_y) = \sum_{z=1}^{n} aff(A_z,A_x)aff(A_z,A_y)$$

Then AM can be written as

$$AM = \sum_{j=1}^{n} [bond(A_j,A_{j-1}) + bond(A_j,A_{j+1})]$$

$$\underbrace{A_1\, A_2\, \ldots\, A_{i-1}}_{AM'}\; A_i\, A_j\; \underbrace{A_{j+1}\, \ldots\, A_n}_{AM''}$$

The global affinity measure for these attributes can be written as

$$AM_{old} = AM' + AM''$$
$$+ bond(A_{i-1}, A_i) + bond(A_i, A_j) + bond(A_j, A_i) + bond(A_j, A_{j+1})$$

- AM_{new}=AM'+AM"+
- $bond(A_{i-1},A_i)+bond(A_i,A_k)+bond(A_k,Ai)+bond(A_k,A_j)+bond(A_j,A_k)+bond(A_j,A_{j-1})$

=>Vậy đóng góp cho AM chung khi đặt Ak vào giữa giữa Ai, Aj,

AMnew-AMold= 2(bond(Ak,Ai)+bond(Ak,Aj)-bond(Ai,Aj))

$$cont(A_i,A_k,A_j) = AM_{new} - AM_{old}$$
$$= 2bond(A_i,A_k) + 2bond(A_k,A_j) - 2bond(A_i,A_j)$$

=>chọn vị trí để đặt Ak để AM max tương đương với Cont (Ai,Ak,Aj) max>0

- Số đo đóng góp của thuộc tính A_k khi đặt vào A_i và A_j (*cont)*
  - A_k, A_i, A_j ; A_i, A_k, A_j ; A_i, A_j, A_k
  - **Điều kiện biên:**

Xếp A_k vào vị trí ngoài cùng bên trái: Thêm cột A_0

Xếp A_k vào vị trí ngoài cùng bên phải: Thêm cột A_n

Các cột A_0 và A_n có các phần tử = 0 trong ma trận lực hút thuộc tính

Thuật toán tụ nhóm BEA (Bond Energy Algorithm)

- Nhóm các thuộc tính của quan hệ toàn cục bằng cách hoán vị các hàng và các cột của ma trận AA, sao cho số đo hấp dẫn ***cont()*** là lớn nhất. Kết quả sẽ là một ma trận tụ hấp dẫn CA (Cluster Affinity).

Thuật toán

- Input: Ma trận *AA*
- Output: Ma trận quan hệ phân cụm *CA*

Bước 1. Khởi tạo: Đặt cột 1 và 2 của AA vào cột 1&2 trong CA.

Bước 2: Giả sử có i cột đã được đặt vào CA. Lấy lần lượt một trong (n-i) cột còn lại của AA, đặt vào cột thứ (i+1) của CA, sao cho số đo AM tại vị trí đó là lớn nhất. Lặp lại bước 2 cho đến hết

Bước 3: Sắp thứ tự hàng theo thứ tự cột

vd

$cont(A_1,A_4,A_2) = 2[bond(A_1, A_4) + bond(A_4,A_2) – bond(A_1,A_2)]$

$bond(A_1, A_4) =$ aff(A1,A1)*aff(A1,A4)+
aff(A2,A1)*aff(A2,A4)+
aff(A3,A1)*aff(A3,A4)+
aff(A4,A1)*aff(A4,A4)

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
Â3	45	5	53	3
A4	0	75	3	78

$bond(A_1, A_4) = 135$
$bond(A_4, A_2) = 11865$
$bond(A_1, A_2) = 225$

$cont(A_1, A_4, A_2) = 2* 135 + 2* 11865 – 2* 225 = 23550$

Giả mã Thuật toán tụ nhóm BEA (Bond Energy Algorithm)

Ví dụ

Chép cột 1 và cột 2 ma trận AA vào ma trận CA

(1) CA(*,1)←AA(*,1)

(2) CA(*,2)←AA(*,2)

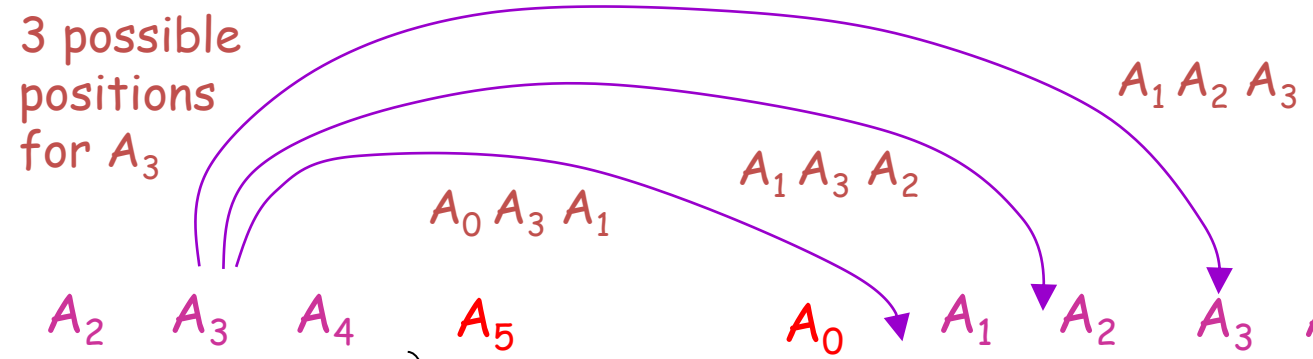
CA =

	A1	A2		
A1	45	0		
A2	0	80		
A3	45	5		
A4	0	75		

AA =

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
A3	45	5	53	3
A4	0	75	3	78

Clustered Affinity Matrix
Step 2: Determine Location for A_3



	A_0	A_1	A_2	A_3	A_4	A_5
A_1	45	0	45	0		
A_2	0	80	5	75		
A_3	45	5	53	3		
A_4	0	75	3	78		

Attribute Affinity Matrix (AA)

	A_0	A_1	A_2	A_3	A_4	A_5
A_1	45	0				
A_2	0	80				
A_3	45	5				
A_4	0	75				

Clustered Affinity Matrix (CA)

Ví dụ

$$bond(A_x,A_y) = \sum_{z=1}^{n} aff(A_z,A_x)aff(A_z,A_y)$$

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
Â3	45	5	53	3
A4	0	75	3	78

index=3

While *index* ≤ *n* **do**

index ≤4 {*thỏa mãn*}

For i **from** 1 **to** *index* – 1 **by** 1 **do**

Tính cont(A_{i-1},A_{index},A_i)

i=1 thứ tự (0-3-1): cont(A_0,A_3,A_1) = 8820

i=2 thứ tự (1-3-2): cont(A_1,A_3,A_2) = 10150

End – for

Điều kiện biên, thứ tự (2-3-4): cont(A_2,A_3,A_4)= 1780

loc =2 thứ tự (1-3-2) có cont =10150lớn nhất

For j **from** *index* **to** *Loc* **by** – 1 **do** {xáo trộn hai ma trận}

CA(*, j) := AA(*,j-1);

CA(*,loc=2):=AA(*,index=3);

Ví dụ

	A1	A3	A2	
A1	45	45	0	
A2	0	5	80	
A3	45	53	5	
A4	0	3	75	

CA =

AA =

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
A3	45	5	53	3
A4	0	75	3	78

Đặt A_3 giữa A_1 và A_2

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
A3	45	5	53	3
A4	0	75	3	78

	A1	A3	A2	
A1	45	45	0	
A2	0	5	80	
A3	45	53	5	
A4	0	3	75	

index=4

While *index* ≤ *n* **do**

index ≤4 *{thỏa mãn}*

For i **from** 1 **to** *index* – 1 **by** 1 **do**

Tính cont(A_{i-1},A_{index},A_i)

i=1 thứ tự (0-4-1): cont(A_0,A_4,A_1) = 270

i=2 thứ tự (1-4-3): cont(A_1,A_4,A_3) = - 7014

i=3 thứ tự (3-4-2): cont(A_3,A_4,A_2) = 23486

End – for

Điều kiện biên, thứ tự (2-4-5): cont(A_2,A_4,A_5)= 23730

loc =4 thứ tự (2-4-5) có cont =23730 lớn nhất

CA =

	A1	A3	A2	A4
A1	45	45	0	0
A2	0	5	80	75
A3	45	53	5	3
A4	0	3	75	78

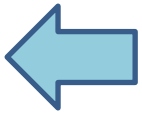
AA =

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
A3	45	5	53	3
A4	0	75	3	78

Đặt A_4 bên phải A_2

CA =

	A1	A3	A2	A4
A1	45	45	0	0
A3	45	53	5	3
A2	0	5	80	75
A4	0	3	75	78



CA =

	A1	A3	A2	A4
A1	45	45	0	0
A2	0	5	80	75
A3	45	53	5	3
A4	0	3	75	78

Chú ý Thuật toán tụ nhóm

- Độ đo cầu nối giữa hai thuộc tính được tính là tổng của tích 2 phần tử cùng hàng của hai cột. Vì ma trận AA đối xứng, có thể thực hiện tương tự theo hàng.
- Trong bước khởi gán, cột 1 và 2 được đặt vào vị trí 1&2 trong CA, vì A2 có thể đặt ở bên trái hoặc phải của A1.
- Nếu A_j là thuộc tính tận trái trong ma trận CA, kiểm tra đóng góp khi đặt thuộc tính A_k vào bên trái của A_j , khi đó bond $(A_{0,} A_k)$ = bond$(A_{0,} A_j)$ =0,
- Nếu A_j là thuộc tính tận phải đã được đặt trong ma trận CA và đang kiểm tra đóng góp khi đặt thuộc tính A_k vào bên phải của A_j, Khi đó bond $(A_{j,} A_{k+1})$ = bond$(A_{k,} A_{k+1})$ =0.

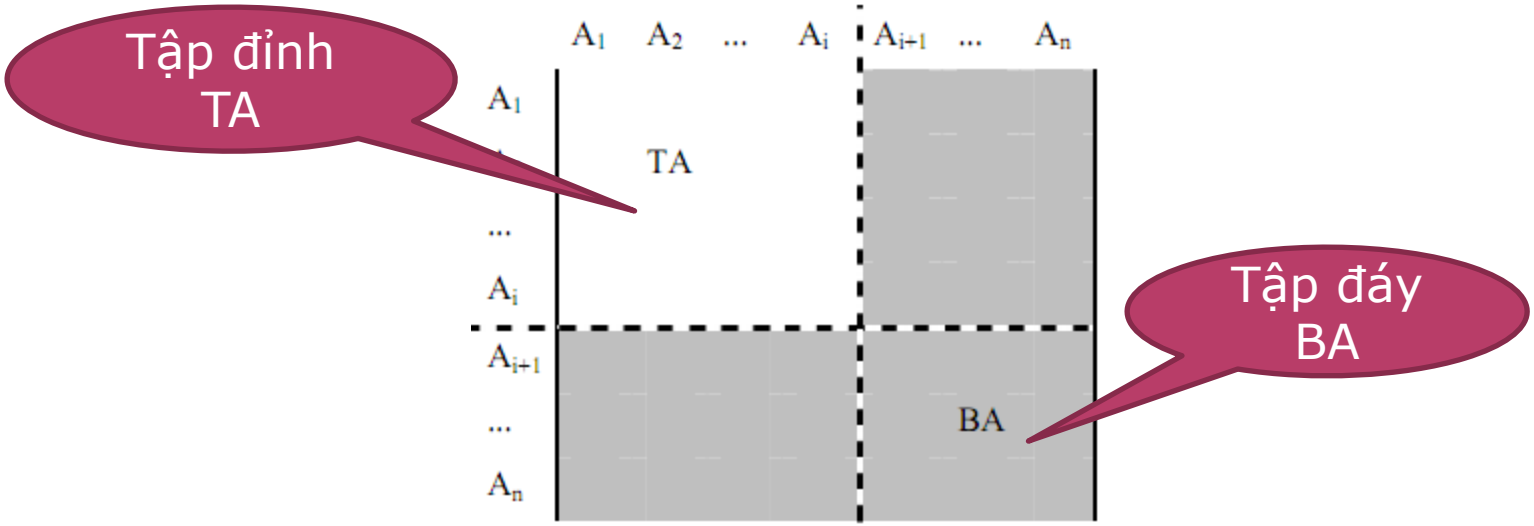
Thuật toán phân mảnh dọc

- Làm thế nào để chia một tập các thuộc tính đã được phân cụm $\{A_1, A_2, \ldots, A_n\}$ thành hai (hoặc nhiều hơn) tập $\{A_1, A_2, \ldots, A_i\}$ và $\{A_{i+1}, \ldots, A_n\}$ sao cho không có (hoặc tối thiểu) ứng dụng nào truy cập cả hai (hoặc nhiều hơn một) tập

Thuật toán phân mảnh dọc

Xét ma trận lực hút tụ nhóm (tụ lực) CA

- TA = {A_1, A_2, …,A_i } ở góc trái cao nhất gọi là tập đỉnh (Top)
- BA = {A_{i+1}, A_{i+2}, …,A_n } ở góc phải thấp nhất gọi là tập đáy (Bottom)



Thuật toán phân mảnh dọc

Ký hiệu	**Ý nghĩa**
$Q = \{q_1, q_2, \ldots, q_n\}$	Tập các ứng dụng.
$AQ(q_i) = \{A_j \mid use(q_i, A_j) = 1\}$	Tập các thuộc tính được truy xuất bởi ứng dụng q_i
$TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$	Tập các ứng dụng chỉ truy xuất trên các thuộc tính TA
$BQ = \{q_i \mid AQ(q_i) \subseteq BA\}$	Tập các ứng dụng chỉ truy xuất trên các thuộc tính BA
$OQ = Q - \{TQ \cup BQ\}$	Tập các ứng dụng truy xuất trên cả BA và TA

Thuật toán phân mảnh dọc

Ký hiệu	Ý nghĩa
$CQ = \sum_{q_i \in \Omega} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$	Tổng chi phí truy xuất của tất cả các ứng dụng trên tất cả các vị trí
$CTQ = \sum_{qi \in TQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$	Tổng chi phí truy xuất của các ứng dụng tới các thuộc tính TA
$CBQ = \sum_{qi \in BQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$	Tổng chi phí truy xuất của các ứng dụng tới các thuộc tính BA
$COQ = \sum_{qi \in OQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$	Tổng chi phí truy xuất của các ứng dụng tới các thuộc tính thuộc cả TA và BA

Thuật toán phân mảnh dọc

Bài toán tối ưu hóa phân mảnh chính là bài toán xác định một điểm : $1 \leq k \leq n$ sao cho :

$$z = CTQ^* CBQ - COQ^2$$ là lớn nhất

Giả mã Thuật toán phân mảnh dọc

Algorithm 3.4: PARTITION Algorithm

Input: CA: clustered affinity matrix; R: relation; ref: attribute usage matrix;
acc: access frequency matrix
Output: F: set of fragments
begin
{determine the z value for the first column}
{the subscripts in the cost equations indicate the split point}
calculate CTQ_{n-1} ;
calculate CBQ_{n-1} ;
calculate COQ_{n-1} ;
$best \leftarrow CTQ_{n-1} * CBQ_{n-1} - (COQ_{n-1})^2$;
repeat
{determine the best partitioning}
for i *from* $n-2$ *to* 1 *by* -1 **do**
calculate CTQ_i ;
calculate CBQ_i ;
calculate COQ_i ;
$z \leftarrow CTQ * CBQ_i - COQ_i^2$;
if $z > best$ **then** $best \leftarrow z$ {record the split point within shift}
call SHIFT(CA)
until *no more SHIFT is possible* ;
reconstruct the matrix according to the shift position ;
$R_1 \leftarrow \Pi_{TA}(R) \cup K$; {K is the set of primary key attributes of R}
$R_2 \leftarrow \Pi_{BA}(R) \cup K$;
$F \leftarrow \{R_1, R_2\}$
end

vdThuật toán phân mảnh dọc

Cách1:

A =

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

A1 A3 A2 A4

CA=

	A1	A3	A2	A4
A1	45	45	0	0
A3	45	53	5	3
A2	0	5	80	75
A4	0	3	75	78

Tập các ưd

AQ(q1)={A1,A3} AQ(q2)={A2,A3}, AQ(q3)={A2,A4}, AQ(q4)={A3,A4}

TA = { A_1} => TQ ={q_i | AQ(q_i) $\subseteq$ TA}= {}},

BA = {A_3, A_2, A_4} =>BQ = {q_i | AQ(q_i) $\subseteq$ BA}={ q_2 , q_3, q_4 } }

OQ ={q_i | AQ(q_i) $\subseteq$ TA}= {q1}, AQ(q_i) $\subseteq$ BA}={q1}}={q1}

Chi phí:

CTQ = 0;

$$Cxx = \sum_{q_i \in xx} \sum_{\forall S_i} ref_i(q_i) \times acc_i(q_i)$$

CBQ = $acc_1(q_2)$ + $acc_2(q_2)$ + $acc_3(q_2)$ +
$acc_1(q_3)$ + $acc_2(q_3)$ + $acc_3(q_3)$ +
$acc_1(q_4)$ + $acc_2(q_4)$ + $acc_3(q_4)$ = 83

COQ = $acc_1(q_1)$ + $acc_2(q_1)$ + $acc_3(q_1)$ = 45

Z = CTQ* CBQ – COQ^2 = -2025

Site1	**Site2**	**Site3**
$acc_1(q_1)$=15	$acc_2(q_1)$=20	$acc_3(q_1)$=10
$acc_1(q_2)$=5	$acc_2(q_2)$=0	$acc_3(q_2)$=0
$acc_1(q_3)$=25	$acc_2(q_3)$=25	$acc_3(q_3)$=25
$acc_1(q_4)$=3	$acc_2(q_4)$=0	$acc_3(q_4)$=0

vdThuật toán phân mảnh dọc

Cách 2:

CA =

	A1	A3	A2	A4
A1	**45**	**45**	0	0
A3	45	53	5	3
A2	0	5	**80**	**75**
A4	0	3	75	78

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

TA = {A_1, A_3 } , TQ = {q_1},

BA = {A_2, A_4} , BQ = {q_3,},

OQ = {q_2, q_4}

$CTQ_2 = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$

$CBQ_2 = acc_1(q_3) + acc_2(q_3) + acc_3(q_3) = 75$

$COQ_2 = acc_1(q_2) + acc_2(q_2) + acc_3(q_2) +$
$acc_1(q_4) + acc_2(q_4) + acc_3(q_4) = 8$

$Z = CTQ^* CBQ - COQ^2 = 3311$

Site1	**Site2**	**Site3**
$acc_1(q_1)=15$	$acc_2(q_1)=20$	$acc_3(q_1)=10$
$acc_1(q_2)=5$	$acc_2(q_2)=0$	$acc_3(q_2)=0$
$acc_1(q_3)=25$	$acc_2(q_3)=25$	$acc_3(q_3)=25$
$acc_1(q_4)=3$	$acc_2(q_4)=0$	$acc_3(q_4)=0$

Vd Thuật toán phân mảnh dọc

CA =

	A1	A3	A2	A4
A1	45	45	0	0
A3	45	53	5	3
A2	0	5	80	75
A4	0	3	75	78

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Cách 3:

TA = $\{A_1, A_3, A_2\}$, TQ = $\{q_2, q_1\}$,

BA = $\{A_4\}$, BQ = { },

OQ = $\{q_4, q_3\}$

$CTQ_3 = acc_1(q_1) + acc_2(q_1) + acc_3(q_1)$

$acc_1(q_2) + acc_2(q_2) + acc_3(q_2) = 50$

$CBQ_3 = 0$

$COQ_3 = acc_1(q_3) + acc_2(q_3) + acc_3(q_3) +$

$acc_1(q_4) + acc_2(q_4) + acc_3(q_4) = 78$

$Z = CTQ^* CBQ - COQ^2 = -6084$

Site1	**Site2**	**Site3**
$acc_1(q_1)=15$	$acc_2(q_1)=20$	$acc_3(q_1)=10$
$acc_1(q_2)=5$	$acc_2(q_2)=0$	$acc_3(q_2)=0$
$acc_1(q_3)=25$	$acc_2(q_3)=25$	$acc_3(q_3)=25$
$acc_1(q_4)=3$	$acc_2(q_4)=0$	$acc_3(q_4)=0$

vdThuật toán phân mảnh dọc

- Cách 1: Z = - 2025
- Cách 2: Z = 3311
- Cách 3: Z = - 6084
- Như vậy cách 2 có chi phí tối ưu nhất
- Quan hệ PROJ chia thành 2 mảnh:

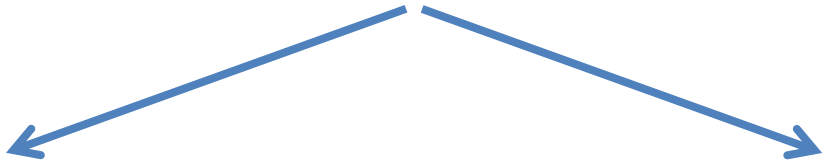
$PROJ_1 \{A_1, A_3\}$ = $PROJ_1 \{\underline{PNO}, BUDGET\}$

$PROJ_2 \{A_1, A_2, A_4\}$ = $PROJ_2 \{\underline{PNO}, PNAME, LOC\}$

vdThuật toán phân mảnh dọc

PROJ

PNO	PNAME	BUDGET	LOG
P1	Instrumentation	150000	Montreal
P2	Database Develop	135000	NewYork
P3	CAD/CAM	250000	NewYork
P4	Maintenance	310000	Paris



PROJ1

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

PROJ2

PNO	PNAME	LOG
P1	Instrumentation	Montreal
P2	Database Develop	NewYork
P3	CAD/CAM	NewYork
P4	Maintenance	Paris

Phân mảnh lai (Hybrid Fragmentation)

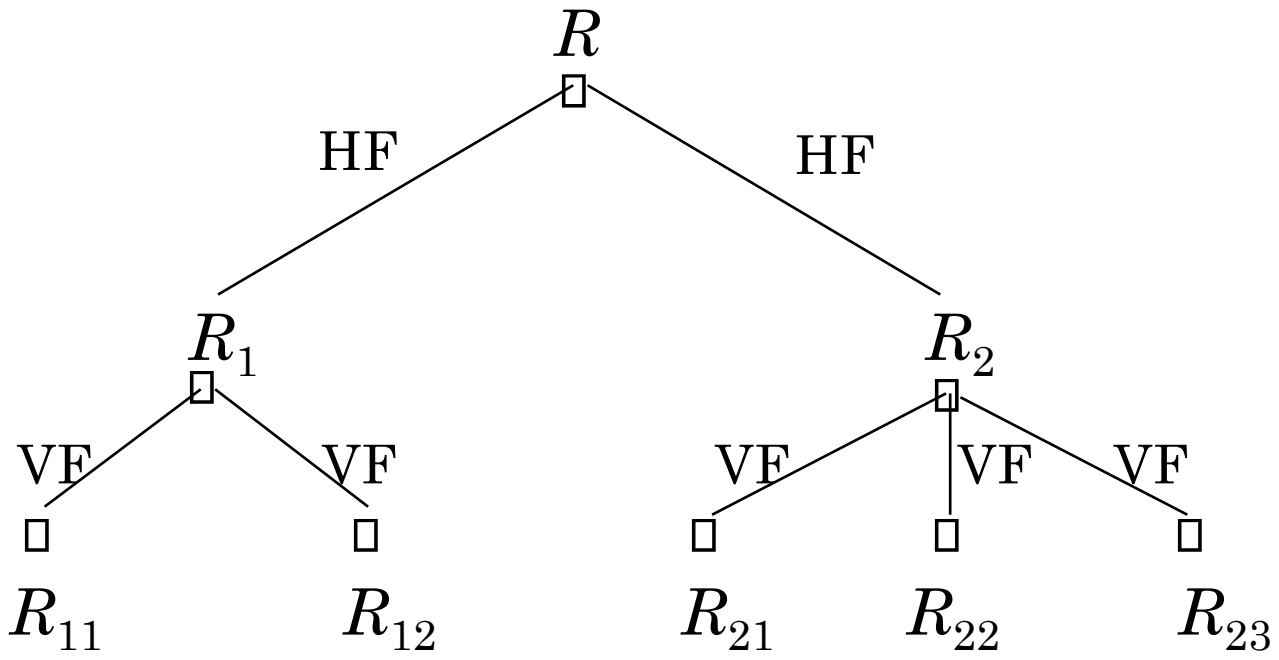
- Sử dụng cả phân mảnh dọc và phân mảnh ngang=> Phân mảnh lai
- Có hai hướng: Phân mảnh ngang rồi đến phân mảnh dọc hoặc ngược lại
- Kiểm tra tính đúng đắn của Phân mảnh lai:

Khôi phục phân đoạn lai

+ VF; Dùng phép kết nối

+ HF: Dùng phép hợp

Phân mảnh lai



Nhân bản và cấp phát dữ liệu (1)

- **Nhân bản**: các mảnh nào sẽ được lưu trữ nhiều bản sao
  - Nhân bản toàn phần
  - Nhân bản có lựa chọn
- **Cấp phát**: Mảnh nào được lưu ở vị trí nào?

Bài toán cấp phát dữ liệu

- Phát biểu bài toán
  - Cho
    - $F = \{F_1, F_2, \ldots, F_n\}$ các mảnh
    - $S = \{S_1, S_2, \ldots, S_m\}$ các vị trí mạng
    - $Q = \{q_1, q_2, \ldots, q_q\}$ các ứng dụng
  - Tìm phân bổ tối ưu của F tới S.
- Tính tối ưu
  - Chi phí tối thiểu
    - Truyền thông + lưu trữ+ xử lý(read & update)
    - Chi phí về thời gian
  - Hiệu năng
    - Thời gian đáp ứng và/hoặc thông lượng
  - Các ràng buộc
    - Các ràng buộc tại mỗi vị trí (lưu trữ & xử lý)

Cấp phát - Yêu cầu thông tin

- Thông tin dữ liệu

  - Chọn lựa các mảnh: Seli(Fj): Số lượng các bộ của Fj đc qi truy xuất xử lý

  - Kích cỡ các mảnh: Size(Fj)= Card(Fj)*length(Fj)

- Thông tin ứng dụng

  - RR_{ij}: số lần truy cập đọc của ứng dụng q_i từ mảnh F_j

  - UR_{ij}: số ghi truy cập nhập (ghi) của ứng dụng q_i đến mảnh F_j

  - u_{ij} một phần tử ma trận chỉ ra truy vấn nào ghi mảnh nào

  - r_{ii} một phần tử ma trận chỉ ra truy vấn nào đọc mảnh nào

$$r_{ij} = \begin{cases} 1 \text{ if query } q_i \text{ retrieves from fragment } F_j \\ 0 \text{ otherwise} \end{cases}$$

- Thông tin vị trí
  - USC_k chi phí đơn vị cho lưu trữ dữ liệu tại vị trí S_k
  - LPC_k chi phí cho xử lý một đơn vị dữ liệu tại vị trí S_k
- Thông tin mạng
  - Chi phí/khung truyền thông giữa hai vị trí Si và Sj: gij
  - Kích cỡ khung: fsize()

Mô hình cấp phát

Dạng tổng quát

min(Tổng chi phí)

đối với

ràng buộc thời gian đáp ứng

ràng buộc lưu trữ

ràng buộc xử lý

- Các biến quyết định

$$x_{ij} = \begin{cases} 1 \text{ nếu mảnh (đoạn) } F_i \text{ được lưu tại vị trí } S_j \\ 0 \text{ trường hợp khác} \end{cases}$$

Cấp phát các mảnh dữ liệu (1)

- Hàm chi phí tổng có hai thành phần: lưu trữ và truy vấn (xl& TT)

$$TOC = \sum_{S_k \in S} \sum_{F_j \in F} STC_{jk} + \sum_{q_i \in Q} QPC_i$$

- Chi phí lưu trữ của mảnh F_j tại vị trí S_k.

$STC_{jk}=USC_k*size(F_j)*X_{jk}$

USC_k là chi phí lưu trữ đơn vị tại vị trí S_k

- Chi phí truy vấn (xl và tt) của một truy vấn q_i bao gồm hai thành phần chi phí xử lý PC và chi phí truyền thông TC $QPC_i = PC_i + TC_i$

Cấp phát các mảnh dữ liệu (2)

- Chi phí xử lý là tổng của 3 thành phần

Chi phí truy cập (AC)(đọc, ghi), chi phí đảm bảo toàn vẹn (IE), chi phí điều khiển tương tranh CC của qi

$$PC_i = AC_i + IE_i + CC_i$$

+ Chi phí truy cập(đọc,ghi)

$$AC_i = \sum_{s_k \in S} \sum_{F_j \in F} (UR_{ij} + RR_{ij}) * x_{ij} * LPC_k$$

LPC_k: Chi phí xử lý đơn vị tại vị trí k

+Chi phí đảm bảo toàn vẹn và điều khiển tương tranh (tính toán tương tự dựa vào các ràng buộc cụ thể)

Cấp phát các mảnh dữ liệu (3)

- **Chi phí truyền dẫn** (Chi phí truyền dẫn cho mỗi truy vấn qi)gồm: hai thành phần: chi phí xử lý cập nhập (ghi) và chi phí xử lý truy vấn (đọc)

$$TC_i = TCU_i + TCR_i$$

Chi phí truyền dẫn cập nhập: chi phí truyền dẫn cho truy vấn cập nhập)

n

$$TCU_i = \sum_{S_k \in S} \sum_{F_j \in F} u_{ij} * (\text{update message cost} + \text{acknowledgment cost})$$

$$TCU_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} * x_{jk} * g_{o(i),k} + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} u_{ij} * x_{jk} * g_{k,o(i)}$$

Chi phí truyền dẫn cho truy vấn

$$TCR_i = \sum_{F_j \in F} \min_{S_k \in S} (x_{jk} * (\text{cost retrieval request} + \text{cost sending back result}))$$

$$TCR_i = \sum_{\forall F_j \in F} \min_{S_k \in S} (r_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk} * \frac{sel_i(F_j) * length(F_j)}{fsize} * g_{k,o(i)})$$

Cấp phát các mảnh dữ liệu (4)

- Mô hình hóa các ràng buộc
  - Ràng buộc thời gian đáp ứng cho truy vấn q_i

  Thời gian thực thi của q_i <= thời gian đáp ứng cực đại cho phép của q_i
  - Các ràng buộc về lưu trữ cho vị trí S_k

$$\sum_{F_j \in F} \text{storage requirement of } F_j \text{ at } S_k \leq \text{ storage capacity of } S_k$$

  - Các ràng buộc về xử lý của vị trí S_k

$$\sum_{q_i \in Q} \text{processing load of } q_i \text{ at site } S_k \leq \text{ processing capacity of } S_k$$

Cấp phát các mảnh dữ liệu (5)

- Độ phức tạp của bài toán cấp phát dữ liệu là NP-complete
- Tương đồng với các bài toán trong lĩnh vực khác
- Bài toán knapsack
- Bài toán luồng mạng
- Do đó, các giải pháp từ các lĩnh vực này có thể được sử dụng
- Sử dụng các kinh nghiệm khác nhau để giảm không gian tìm kiếm
  - Giả sử rằng tất cả các phân chia ứng cử được xác định cùng nhau với các chi phí và các lời ích liên quan ở góc độ xử lý truy vấn
  - Vấn đề được rút gọn thành việc tìm phân chia tối ưu và vị trí cho mỗi quan hệ
  - Bỏ qua việc nhân bản ở bước đầu tiên và tìm giải pháp tối ưu cho bài toán trong trường hợp không nhân bản
  - Vấn đề nhân bản sau đó được xử lý ở bước thứ 2

Tổng kết

- Vấn đề thiết kế
- Các chiến lược thiết kế (trên xuống, dưới lên)
- Phân mảnh dữ liệu (ngang, đứng)
- Cấp phát và nhân bản các phân mảnh