Lecture 2: DDBS Design

Objectives:

- Design problem
- Design strategies (top-down, bottom-up)
- Fragmentation (horizontal, vertical)
- Allocation and replication of fragments, optimality, heuristics

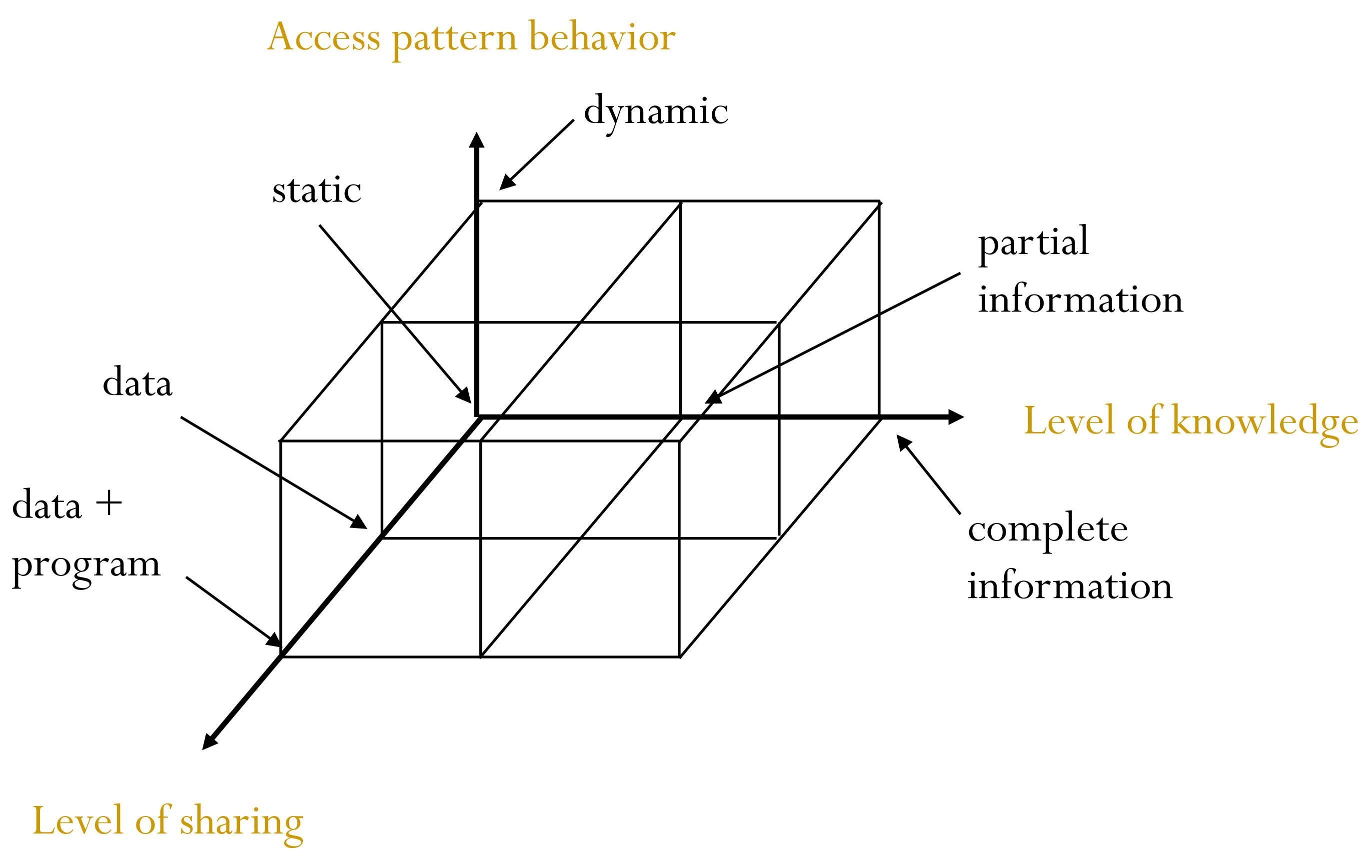
Hoa Dinh Nguyen, PhD.
Department of Information Technology

Design Issues

- General setting:

  Making decisions about the placement of data and programs across the sites of a computer network as well as possibly designing the network itself
- In Distributed DBMS, the placement of applications entails
  - Placement of the distributed DBMS software;
  - Placement of the applications that run on the database

Designing Dimensions



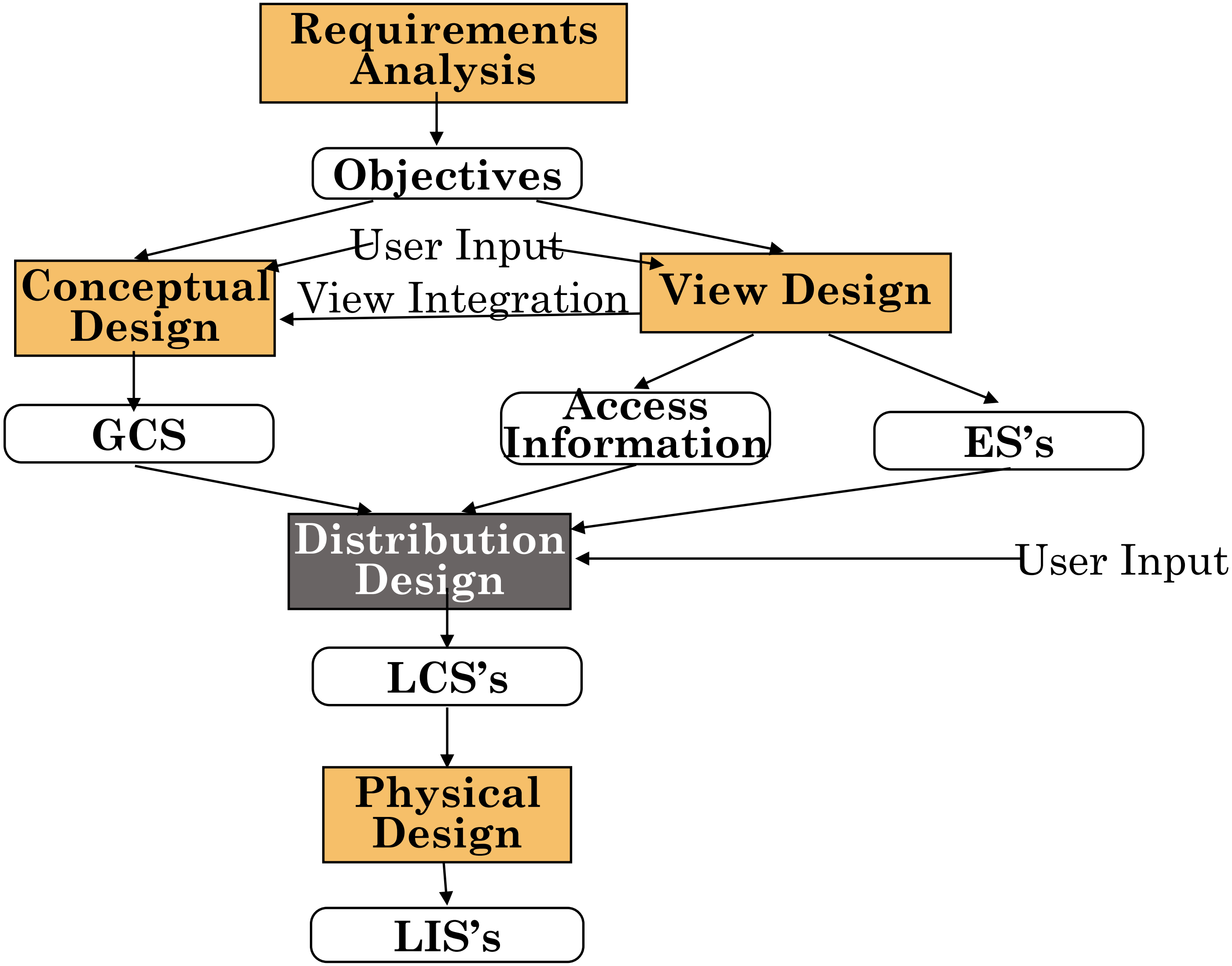
Design Approaches

www.ptit.edu.vn

- Top-down approach
  - Designing systems from scratch
  - Homogeneous systems
- Bottom-up approach
  - The databases already exist at a number of sites
  - The databases should be connected to solve common tasks

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Top-down Approach



Distribution Design Strategies

- Objective: Design the LCSs by distributing the entities (relations) over the sites
- Two main aspects have to be designed carefully
  - **Fragmentation**: Relation may be divided into a number of sub-relations, which are distributed
  - **Allocation and Replication**: Each fragment is stored at site with ”optimal” distribution

Distribution Design Issues

- Why to do fragmentation?
- How to fragmentize?
- How much to Fragmentize?
- How to test correctness?
- How to allocate?
- Information requirements?

Fragmentation Aims

- Reliability
- Performance
- Balanced storage capacity and costs
- Communication costs
- Security

Data Fragmentation

What is a reasonable unit of distribution? Relation or fragment of relation?

Relations as unit of distribution:

- If the relation is not replicated, we get a high volume of remote data accesses
- If the relation is replicated, we get unnecessary replications, which cause problems in executing updates and waste disk space
- Might be an OK solution, if queries need all the data in the relation and data stays only at the sites that use the data

Data Fragmentation

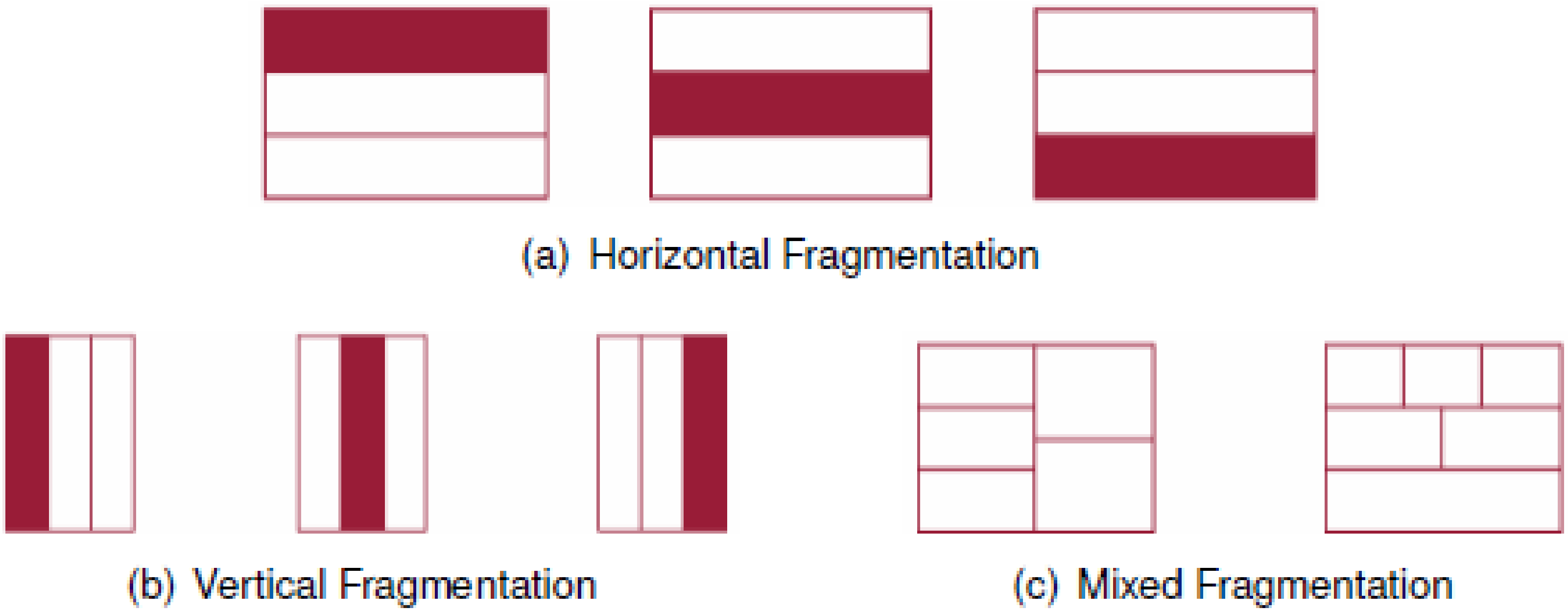
Fragments of relation as unit of distribution

- Application views are usually subsets of relations. Thus, locality of accesses of applications is defined on subsets of relations
- Permits a number of transactions to execute concurrently, since they will access different portions of a relation
- Parallel execution of a single query (intra-query concurrency)
- However, semantic data control (especially integrity enforcement) is more difficult

=> Fragments of relations are (usually) appropriate unit of distribution

Types of Fragmentation

Horizontal: partitions a relation along its tuples
Vertical: partitions a relation along its attributes
Mixed/hybrid: a combination of horizontal and vertical fragmentation



Hoa Dinh Nguyen, PhD.
Department of Information Technology

Horizontal Fragmentation

$PROJ_1$: Projects with budgets smaller than \$200,000

$PROJ_2$: Projects with budgets greater than or equal to \$200,000

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	15000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Vertical Fragmentation

$PROJ_1$: Information on projects' budgets

$PROJ_2$: Information on names and locations

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

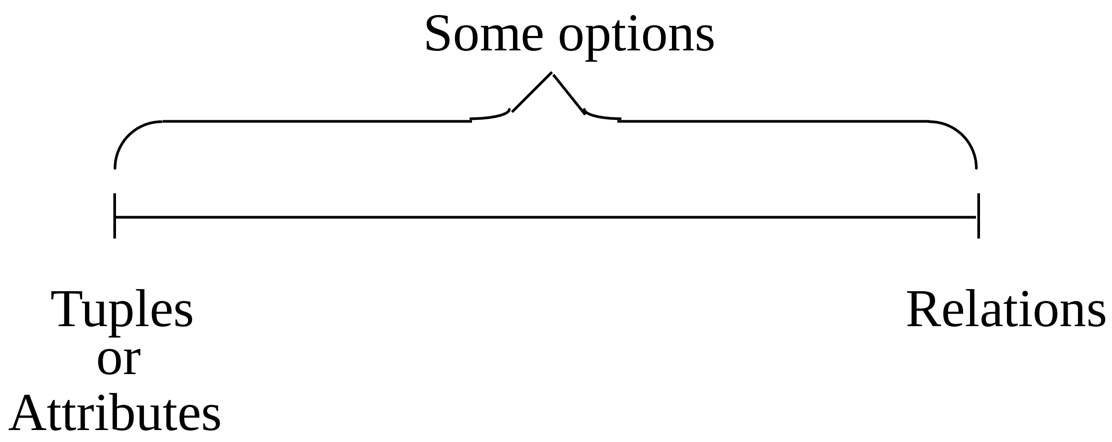
$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris
P5	CAD/CAM	Boston

Degree of Fragmentation



Characteristics of Fragmentation

Completeness

- Decomposition of relation R into fragments R_1, R_2, ..., R_n is complete if and only if each data item in R can also be found in some R_i

Reconstruction

- If relation R is decomposed into fragments R_1, R_2, ..., R_n, then there should exist some relational operator ∇ such that

$$R = \nabla_{1 \le i \le n} R_i \ \nabla$$

Disjointness

- If relation R is decomposed into fragments R_1, R_2, ..., R_n, and data item d_i is in R_j, then d_i should not be in any other fragment R_k $(k \neq j)$.

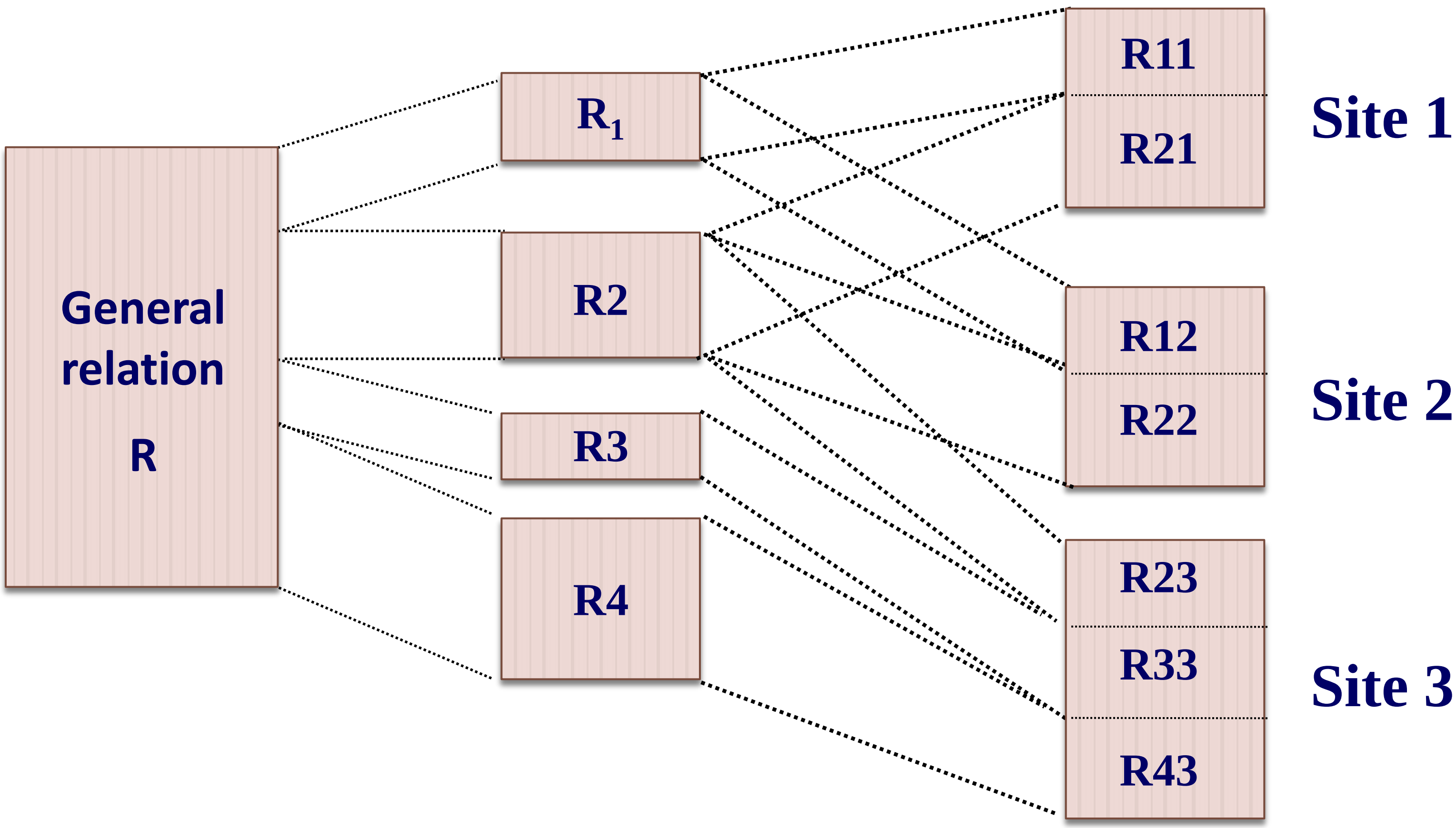
Hoa Dinh Nguyen, PhD.
Department of Information Technology

Allocation methods

- Non-replicated
  - partitioned : each fragment resides at only one site
- Replicated
  - fully replicated : each fragment at each site
  - partially replicated : each fragment at some of the sites
- Rule of thumb:
  - If $\frac{\text{read-only queries}}{\text{update quries}} \geq 1$, then replication is needed
  - Otherwise, replication is not good

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Allocation



General relation R
R1
R2
R3
R4
R11
R21
R12
R22
R23
R33
R43
Site 1
Site 2
Site 3

Allocation

- General relation R is fragmented into 4 fragments R_1, R_2, R_3, R_4, located in 3 sites
  - Site 1: a replication of R_1 (R_{11}) and a replication of R_2 (R_{21})
  - Site 2: a replication of R_1 (R_{12}) and a replication of R_2 (R_{22})
  - Site 3: replications of R_2 (R_{23}), R_3 (R_{33}) and R_4 (R_{43})

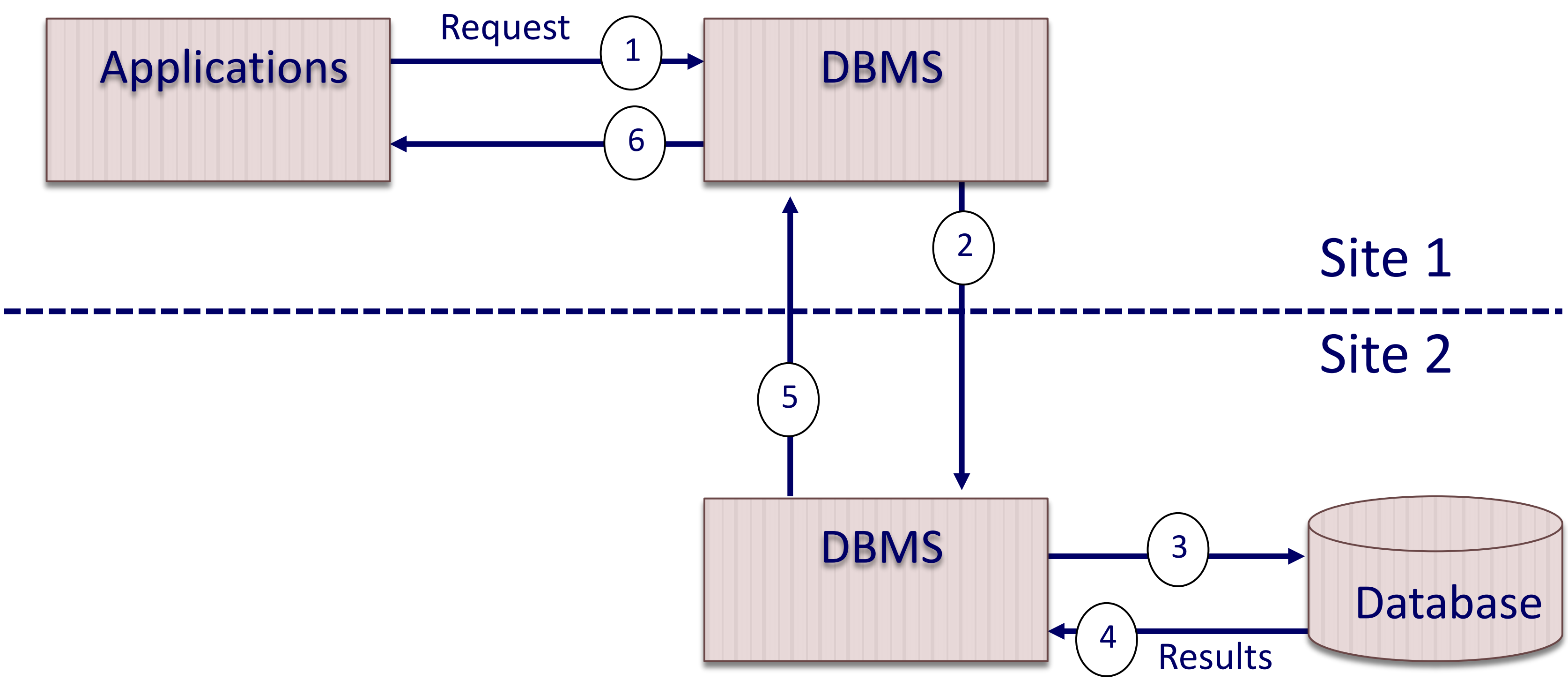
Hoa Dinh Nguyen, PhD.
Department of Information Technology

Fragmentation and Allocation

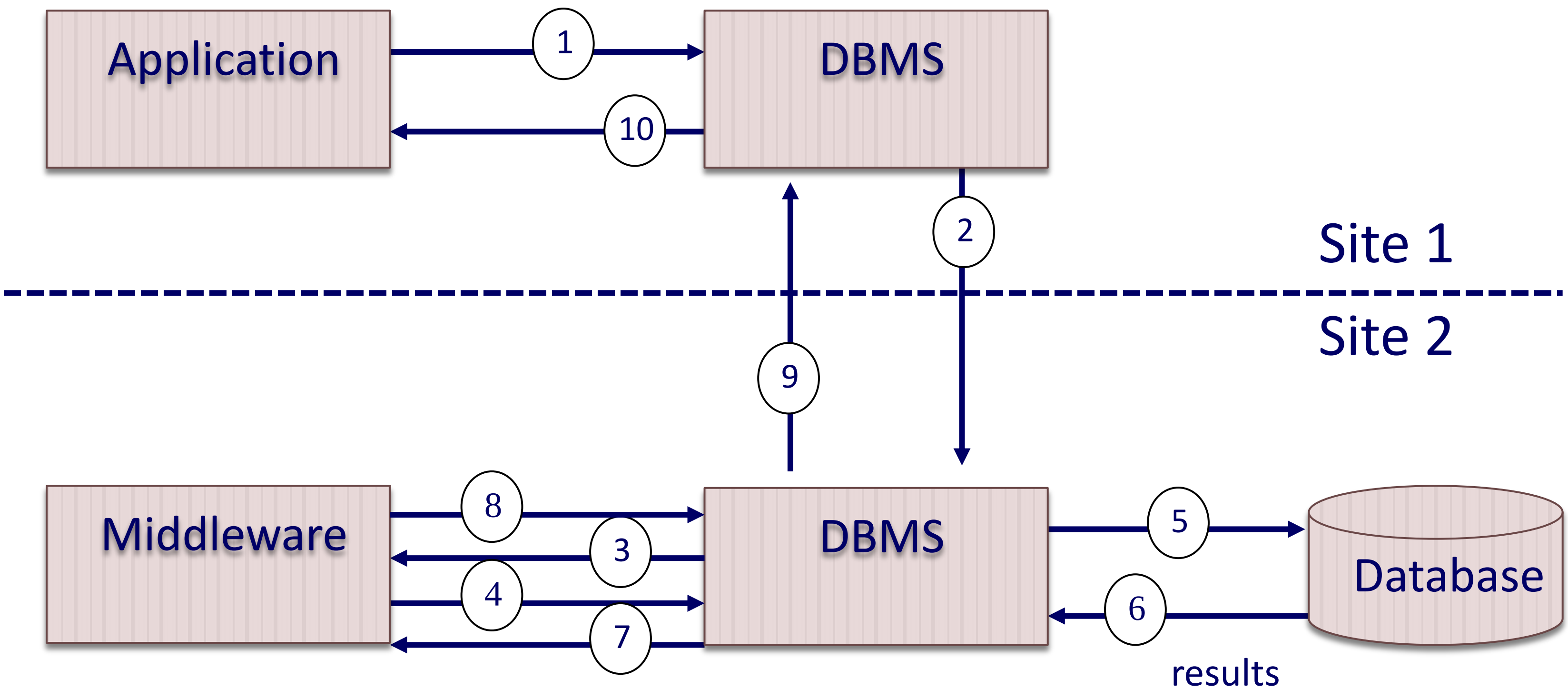
- Allocation is transparent to users, at a lower transparency level than fragmentation
- Users work on local fragments, instead of general relation. Users has no idea where the data is located
- Fragmentation is different from allocation.

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Direct Remote Access



Direct Remote Access via Middleware



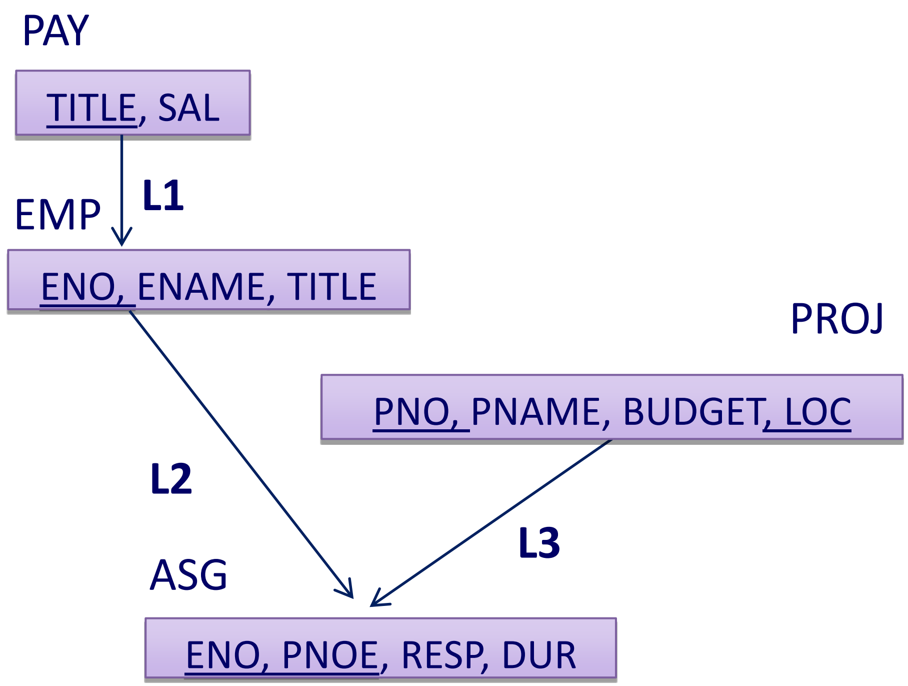
Information Requirements

- Four categories:
  - Database information
  - Application information
  - Communication network information
  - Computer system information

Information Requirements (1)

Database information

- Relationship between relations



- Cardinality of each relation: card(R)

Information Requirements (2)

Application information

- **simple predicates :** Given $R[A_1, A_2, \ldots, A_n]$, a simple predicate p_j is

  $p_j : A_i\ \theta\ Value$

  where $\theta \in \{=,<,\leq,>,\geq,\neq\}$, $Value \in D_i$ and D_i is a domain of A_i.

  For a relation R we define $Pr = \{p_1, p_2, \ldots, p_m\}$

  Example :

  PNAME = "Maintenance"

  BUDGET $\leq$ 200000

- **Minterm predicates**: Given R and $Pr=\{p_1, p_2, \ldots, p_m\}$

  Define $M=\{m_1, m_2, \ldots, m_r\}$ as:

  $$M=\{\ m_i | m_i = \wedge_{p_j \in Pr} p_j^*\ \},\ 1 \leq j \leq m,\ 1 \leq i \leq z$$

  where $p_j^* = p_j$ or $p_j^* = \neg(p_j)$.

Information Requirements (3)

Example:

m_1: PNAME="Maintenance"$\wedge$ BUDGET$\leq$200000

m_2: **NOT**(PNAME="Maintenance")$\wedge$ BUDGET$\leq$200000

m_3: PNAME= "Maintenance"$\wedge$ **NOT**(BUDGET$\leq$200000)

m_4: **NOT**(PNAME="Maintenance")$\wedge$ **NOT**(BUDGET$\leq$200000)

Information Requirements (3)

Example:

p1: TITLE = "Elect.Eng"

p2: TITLE = "Syst. Anal"

p3: TITLE = "Mech. Eng"

p4: TITLE = "Programmer"

p5: SAL $\leq$ 30000

p6: SAL > 30000

PAY

TITLE	SAL
Elect.Eng	40000
Mech.Eng	27000
Programmer	24000
Syst.Anal	34000

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Information Requirements (3)

m1: $TITLE = \text{“Elect.Eng”} \wedge SAL \leq 30000$

m2: $TITLE = \text{“Elect.Eng”} \wedge SAL > 30000$

m3: $\neg(TITLE = \text{“Elect.Eng”}) \wedge SAL \leq 30000$

m4: $\neg(TITLE = \text{“Elect.Eng”}) \wedge SAL > 30000$

m5: $TITLE = \text{“Programmer”} \wedge SAL \leq 30000$

m6: $TITLE = \text{“Programmer”} \wedge SAL > 30000$

Information Requirements (4)

Application information (cont.)

- Minterm selection: $sel(m_i)$
  - Some tuples of the relation retrieved by a query is defined by a minterm predicate m_i
- Access frequency: $acc(q_i)$
  - Data access frequency of an application q_i
  - Access frequency of a minterm predicate can also be defined.

Primary Horizontal Fragmentation (PHF)

Definition :

$$R_j = \sigma_{F_j}(R), \; 1 \le j \le w$$

where F_j is a selection formula, which is (preferably) a minterm predicate.

Therefore,

A horizontal fragment R_i of relation R consists of all the tuples of R which satisfy a minterm predicate m_i.

Given a set of minterm predicates M, there are as many horizontal fragments of relation R as there are minterm predicates.

Set of horizontal fragments also referred to as *minterm fragments.*

Hoa Dinh Nguyen, PhD.
Department of Information Technology

PHF - Example

Given a set of minterms:

m1: $\{BUDGET \le 200000\}$

m2: $\{200000 < BUDGET \le 400000\}$

m3: $\{400000 < BUDGET \le 600000\}$

m4: $\{600000 < BUDGET\}$

Relation PROJ is fragmented into

$PROJ1 = \sigma_{BUDGET \le 200000}(PROJ)$

$PROJ2 = \sigma_{200000 < BUDGET \le 400000}(PROJ)$

$PROJ3 = \sigma_{400000 < BUDGET \le 600000}(PROJ)$

$PROJ4 = \sigma_{600000 < BUDGET}(PROJ)$

PHF - Algorithm

- Given: A relation *R*, and the set of simple predicates *Pr*
- Output: The set of fragments of $R = \{R_1, R_2, \ldots, R_w\}$ which obey the fragmentation rules
- Preliminaries :
  - *Pr* should be complete
  - *Pr* should be minimal

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Completeness of Simple Predicates (1)

- A set of simple predicates *Pr* is said to be *complete* if and only if the accesses to the tuples of the minterm fragments defined on *Pr* requires that two tuples of the same minterm fragment have the same probability of being accessed by any application
- Example:
  - Assume PROJ[PNO,PNAME,BUDGET,LOC] has two applications defined on it
    - Find the budgets of projects at certain locations, (1)
    - Find projects with budgets less than $200000. (2)

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Completeness of Simple Predicates(2)

According to (1),

Pr={LOC="Montreal",LOC="New York",LOC="Paris"}

which is not complete with respect to (2).

Modify

Pr ={LOC="Montreal",LOC="New York",LOC="Paris", BUDGET≤200000,BUDGET>200000}

which is complete

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Minimal of Simple Predicates (1)

- If a predicate influences how fragmentation is performed, (i.e., causes a fragment f to be further fragmented into, say, f_i and f_j) then there should be at least one application that accesses f_i and f_j differently
- In other words, the simple predicate should be relevant in determining a fragmentation
- If all the predicates of a set Pr are relevant, then Pr is minimal

$$\frac{acc(m_i)}{card(f_i)} \neq \frac{acc(m_j)}{card(f_j)}$$

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Minimal of Simple Predicates (2)

Example:

Pr ={LOC="Montreal",LOC="New York", LOC="Paris",

BUDGET≤200000,BUDGET>200000}

is minimal (in addition to being complete).

However, if we add

PNAME = "Instrumentation"

then Pr is not minimal.

COM_MIN Algorithm (1)

- Input: a relation *R* and a set of simple predicates *Pr*
- Output: a complete and minimal set of simple predicates *Pr'* for *Pr*
- *Rule 1*: a relation or fragment is partitioned into at least two parts which are accessed differently by at least one application.

Hoa Dinh Nguyen, PhD.
Department of Information Technology

COM_MIN Algorithm (2)

- **Initialization**:
  - Find a $p_i \in Pr$ such that p_i fragmentize R following rule 1
  - Set $Pr' = p_i$; $Pr \leftarrow Pr - p_i$; $F \leftarrow f_i$
- **Repeat** adding predicates to Pr' until finish.
  - Find a $p_j \in Pr$ such that p_j fragmentize a f_k defined by a minterm predicate on Pr' following rule 1
  - Set $Pr' = Pr' \cup p_i$; $Pr \leftarrow Pr - p_i$; $F \leftarrow F \cup f_i$
  - If $\exists\, p_k \in Pr'$ and is not relevant then
    - $Pr' \leftarrow Pr' - p_k$
    - $F \leftarrow F - f_k$

www.ptit.edu.vn

Hoa Dinh Nguyen, PhD.
Department of Information Technology

PHORIZONTAL Algorithm

- Makes use of COM_MIN to perform fragmentation
- Input: a relation R and a set of simple predicates Pr
- Output: a set of minterm predicates M according to which relation R is to be fragmented

- $Pr' \leftarrow$ COM_MIN(R,Pr)
- determine the set M of minterm predicates
- determine the set I of implications among $p_i \in Pr$
- eliminate the contradictory minterms from M

PHF- Example

- Two candidate relations: PAY and PROJ.
- Fragmentation of relation PAY
  - Application: Check the salary info and determine raise
  - Employee records kept at two sites
  - Applications run at two sites
  - Simple predicates
    - p_1 : SAL $\leq$ 30000
    - p_2 : SAL $>$ 30000
    - $Pr' = \{p_1\}$ which is complete and minimal
  - Minterm predicates
    - m_1 : (SAL $\leq$ 30000)
    - m_2 : **NOT**(SAL $\leq$ 30000) = (SAL $>$ 30000)

PHF-Example

PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

PHF- Example

- Fragmentation of relation PROJ
  - Applications:
    - Find the name and budget of projects given their number is issued at three sites
    - Access project information according to budget
      - one site accesses ≤ 200000 others access > 200000
  - Simple predicates
  - For application (1)
    - p_1 : LOC = "Montreal"
    - p_2 : LOC = "New York"
    - p_3 : LOC = "Paris"
  - For application (2)
    - p_4 : BUDGET ≤ 200000
    - p_5 : BUDGET > 200000
  - $Pr = Pr' = \{p_1, p_2, p_3, p_4, p_5\}$

PHF- Example

- Fragmentation of relation PROJ (cont.)
  - Minterm fragments left after elimination

$$m_1 : (\text{LOC} = \text{“Montreal”}) \wedge (\text{BUDGET} \leq 200000)$$

$$m_2 : (\text{LOC} = \text{“Montreal”}) \wedge (\text{BUDGET} > 200000)$$

$$m_3 : (\text{LOC} = \text{“New York”}) \wedge (\text{BUDGET} \leq 200000)$$

$$m_4 : (\text{LOC} = \text{“New York”}) \wedge (\text{BUDGET} > 200000)$$

$$m_5 : (\text{LOC} = \text{“Paris”}) \wedge (\text{BUDGET} \leq 200000)$$

$$m_6 : (\text{LOC} = \text{“Paris”}) \wedge (\text{BUDGET} > 200000)$$

PHF – Correctness

- Completeness
  - Since *Pr'* is complete and minimal, the selection predicates are complete
- Reconstruction
  - If relation R is fragmented into $F_R = \{R_1, R_2, \ldots, R_r\}$

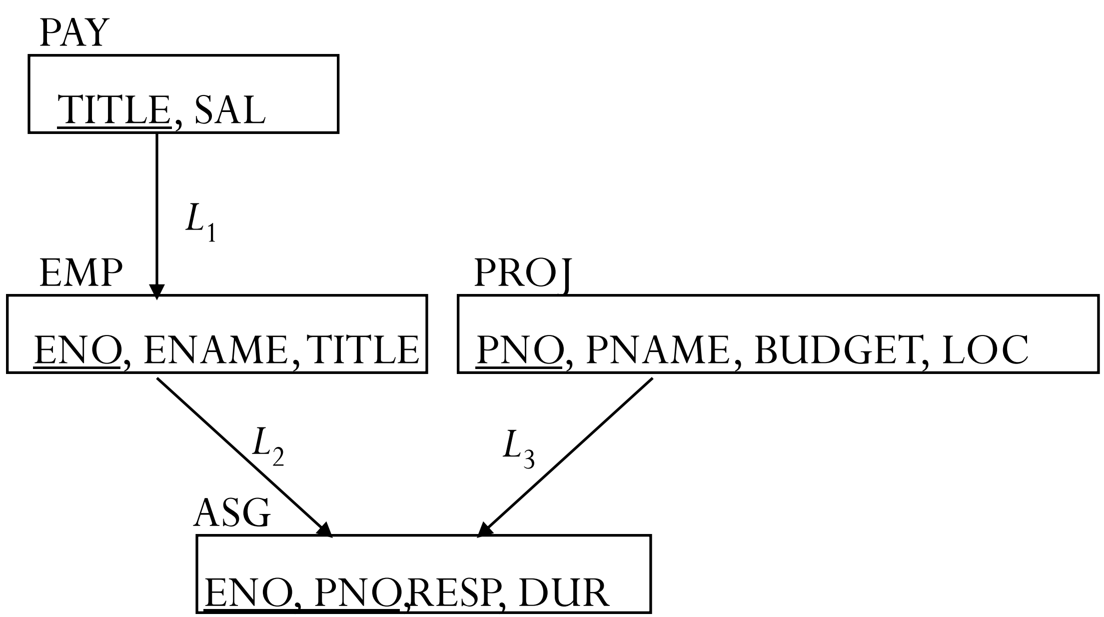
$$R = \bigcup_{\forall R_i \in F_R} R_i$$

- Disjointness
  - Minterm predicates that form the basis of fragmentation should be mutually exclusive

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Derived Horizontal Fragmentation

- Defined on a member relation of a link according to a selection operation specified on its owner
  - Each link is an equijoin
  - Equijoin can be implemented by means of semijoins



DHF- Definition

Given a link L where $owner(L)=S$ and $member(L)=R$, the derived horizontal fragments of R are defined as:

$$R_i = R \ltimes S_i,\ 1 \leq i \leq w$$

Where w is the maximum number of fragments that will be defined on R and

$$S_i = \sigma_{F_i}(S)$$

Where F_i is the formula according to which the primary horizontal fragment S_i is defined.

Hoa Dinh Nguyen, PhD.
Department of Information Technology

DHF – Example

Given a link L_1 where *owner*(L_1)=SKILL and *member*(L_1)=EMP

$$EMP_1 = EMP \ltimes SKILL_1$$
$$EMP_2 = EMP \ltimes SKILL_2$$

where

$$SKILL_1 = \sigma_{SAL \leq 30000}(PAY)$$
$$SKILL_2 = \sigma_{SAL > 30000}(PAY)$$

EMP_1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

EMP_2

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Syst. Anal.

DHF – Correctness

- Completeness
  - Referential integrity
  - Let R be the member relation of a link whose owner is relation S which is fragmented as $F_S = \{S_1, S_2, ..., S_n\}$. Let A be the join attribute between R and S. Then, for each tuple t of R, there should be a tuple t' of S such that

$$t[A]=t'[A]$$

- Reconstruction
  - Same as primary horizontal fragmentation
- Disjointness
  - Simple join graphs between the owner and the member fragments

Vertical Fragmentation (1)

Objective of vertical fragmentation is to partition a relation into a set of smaller relations so that many of the applications will run on only one fragment.

- Vertical fragmentation of a relation *R* produces fragments R_1, R_2…, each of which contains a subset of *R*'s attributes.
- Vertical fragmentation is defined using the projection operation of the relational algebra:

$$\pi_{A1,A2,\ldots Ak}(R)$$

- Example:
  - PROJ1 = $\pi_{PNO,BUDGET}$ (PROJ)
  - PROJ2 = $\pi_{PNO,\ PNAME,\ LOC}$ (PROJ)
- Vertical fragmentation has also been studied for (centralized) DBMS to create Smaller relations, and hence less page accesses
- Vertical fragmentation is more complicated than horizontal fragmentation

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Vertical Fragmentation (2)

Two types of heuristics for vertical fragmentation:

- **Grouping:** assign each attribute to one fragment, and at each step, join some of the fragments until some criteria is satisfied
  - Bottom-up approach
- **Splitting**: starts with a relation and decides on beneficial partitionings based on the access behaviour of applications to the attributes
  - Top-down approach
  - Results in non-overlapping fragments
- Replication of key attributes is not considered as overlapping
- Optimal solution is probably closer to the full relation than to a set of small relations with only one attribute

Hoa Dinh Nguyen, PhD.
Department of Information Technology

VF – Application Information

- Attribute affinities
  - A measure that indicates how closely related the attributes are
  - This is obtained from more primitive usage data
- Attribute usage values
  - Given a set of queries $Q = \{q_1, q_2, \ldots, q_q\}$ that will run on the relation $R[A_1, A_2, \ldots, A_n]$,

$$use(q_i, A_j) = \begin{cases} 1 & \text{if feature } A_j \text{ is used by } q_i \\ 0 & \text{otherwise} \end{cases}$$

- Vector use(qi,•) can be defined accordingly

VF – Application Information

- Use matrix

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)

Hoa Dinh Nguyen, PhD.
Department of Information Technology

VF-Valued-based Matrix

Consider a set of 4 applications on the relation PROJ(PNO, PNAME, BUDGET, LOC)

- Find budgets of projects with certain project numbers
- Find names and budgets of all projects
- Find names of projects in some certain locations
- Find total budget of all projects in each city.

Hoa Dinh Nguyen, PhD.
Department of Information Technology

VF-Valued-based Matrix

Consider the following 4 queries for relation PROJ

q_1: **SELECT** BUDGET
FROM PROJ
WHERE PNO=Value

q_2: **SELECT** PNAME,BUDGET
FROM PROJ

q_3: **SELECT** PNAME
FROM PROJ
WHERE LOC=Value

q_4: **SELECT** **SUM**(BUDGET)
FROM PROJ
WHERE LOC=Value

Let abbreviate A_1= PNO, A_2= PNAME, A_3= BUDGET, A_4= LOC

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

VF - Affinity Measure $aff(A_i,A_j)$

- $aff(A_i,A_j)$ defines the frequency that attributes A_i and A_j co-exist in queries

$$Q = (q_1, q_2, \ldots, q_n)$$

$$aff(A_i, A_j) = \sum_{\substack{k:\ use(q_k, A_i)=1, \\ use(q_k, A_j)=1}} \left(\sum_{\text{sites } l} ref_l(q_k) * acc_l(q_k) \right)$$

- Where
  - $ref_l(q_k)$ is the cost (= number of accesses to (A_i,A_j)) of query q_k at site l)
  - $acc_l(q_k)$ is the frequency of query q_k at site l

VF - Affinity Measure *aff*(A_i,A_j)

- 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)

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Example- Affinity Matrix

- Assume that the costs of the query: $ref_I(q_k) = 1$

$$\begin{array}{c|cccc} & A_1 & A_2 & A_3 & A_4 \\ \hline q_1 & 1 & 0 & 1 & 0 \\ q_2 & 0 & 1 & 1 & 0 \\ q_3 & 0 & 1 & 0 & 1 \\ q_4 & 0 & 0 & 1 & 1 \end{array}$$

- Frequency of the query at sites: $acc_I(q_k)$

$$\begin{array}{c|ccc} & S_1 & S_2 & S_3 \\ \hline q1 & 15 & 20 & 10 \\ q2 & 5 & 0 & 0 \\ q3 & 25 & 25 & 25 \\ q4 & 3 & 0 & 0 \end{array}$$

- $aff(A_1, A_3) = 15*1 + 20*1+10*1 = 45$

Attribute relationship matrix *AA*

$$\begin{array}{c|cccc} & A_1 & A_2 & A_3 & A_4 \\ \hline A_1 & 45 & 0 & 45 & 0 \\ A_2 & 0 & 80 & 5 & 75 \\ A_3 & 45 & 5 & 53 & 3 \\ A_4 & 0 & 75 & 3 & 78 \end{array}$$

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Example – Affinity Matrix

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

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

VF – Clustering Algorithm

- Idea: Take the attribute affinity matrix *AA* and reorganize the attribute orders to form clusters where the attributes in each cluster demonstrate high affinity to one another
- Bond Energy Algorithm (BEA) – has been used for clustering of entities. BEA finds an ordering of entities (in our case, attributes) such that the global affinity measure

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

is maximum

Hoa Dinh Nguyen, PhD.
Department of Information Technology

BEA Algorithm (1)

- Input: The *AA* matrix
- Output: The clustered affinity matrix *CA* which is a perturbation of *AA*
- **Initialization**: Place and fix one of the columns of *AA* in *CA*
- **Iteration**: Place the remaining *n-i* columns in the remaining *i+1* positions in the *CA* matrix. For each column, choose the placement that makes the most contribution to the global affinity measure
- **Row order**: Order the rows according to the column ordering

Hoa Dinh Nguyen, PhD.
Department of Information Technology

BEA Algorithm (2)

Best placement? Definition of bond placement of attribute A_k:

$$cont(A_i, A_k, A_j) = 2bond(A_i, A_k) + 2bond(A_k, A_l) - 2bond(A_i, A_j)$$

Where the constraint

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

Place A_k at the far left position: add column A_0

Place A_k at the far right position: add column A_n

Columns A_0 and A_n have zero-components in the affinity matrix

Hoa Dinh Nguyen, PhD.
Department of Information Technology

BEA - Example (1)

Consider matrix AA and a corresponding CA where A_1 and A_2 are placed. Place A_3:

$$AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{bmatrix} \end{array} \qquad CA = \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \quad & \quad \end{array} \\ \begin{bmatrix} 45 & 0 & \quad & \quad \\ 0 & 80 & & \\ 45 & 5 & & \\ 0 & 75 & & \end{bmatrix} \end{array}$$

Order (0-3-1) :

$$cont(A_0,A_3,A_1) = 2bond(A_0 , A_3)+2bond(A_3 , A_1)–2bond(A_0 , A_1)$$
$$= 2*0 + 2*4410 – 2*0 = 8820$$

Order(1-3-2) :

$$cont(A_1,A_3,A_2) = 2bond(A_1 , A_3)+2bond(A_3 , A_2)–2bond(A_1,A_2)$$
$$= 2*4410 + 2*890 – 2*225 = 10150$$

Order (2-3-4) :

$$cont(A_2,A_3,A_4) = 1780$$

BEA-Example (2)

- Placement of A_1, A_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

BEA-Example (3)

- Placement of A_3

CA =

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

BEA-Example (4)

- Placement of A_4

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

BEA - Example (5)

- When A_4 has been placed => Matrix CA?

CA =

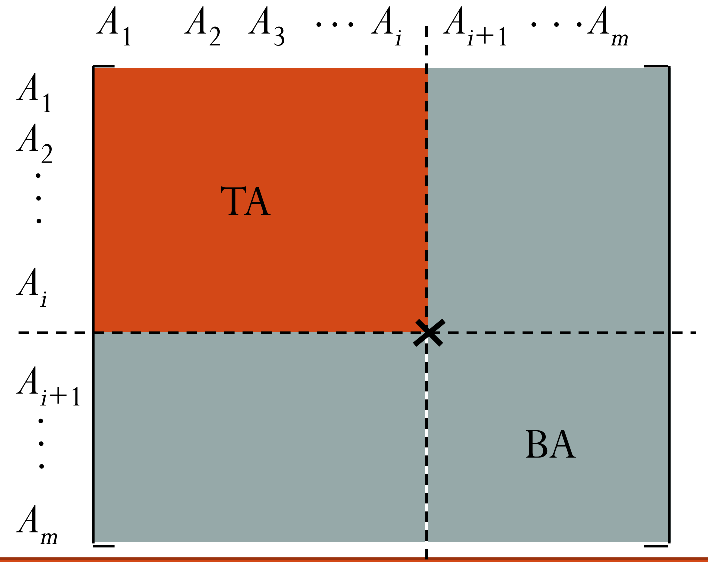
	A1	A3	A2	A4
A1	45	45	0	0
A3	45	53	5	3
A2	0	5	80	75
A4	0	3	75	78

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

VF- Division Algorithm (1)

- How can you divide a set of clustered attributes $\{A_1, A_2, \ldots, A_n\}$ into two (or more) sets $\{A_1, A_2, \ldots, A_i\}$ and $\{A_i, \ldots, A_n\}$ such that there are no (or minimal) applications that access both (or more than one) of the sets?



Hoa Dinh Nguyen, PhD.
Department of Information Technology

VF – Division Algorithm (2)

Notation	**Meaning**
$Q = \{q_1, q_2, \ldots, q_n\}$	Set of applications
$AQ(q_i) = \{A_j \mid use(q_i, A_j) = 1\}$	Set of attributes accessed by application q_i
$TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$	set of applications that access only TA
$BQ = \{q_i \mid AQ(q_i) \subseteq BA\}$	set of applications that access only BA
$OQ = Q - \{TQ \cup BQ\}$	set of applications that access both BA and TA
$CQ = \sum_{q_i \in \Omega} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$	Total costs of all applications in all sites

VF – Division Algorithm (3)

Define

CTQ = total number of accesses to attributes by applications that access only TA

$$CTQ = \sum_{qi \in TQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$$

CBQ = total number of accesses to attributes by applications that access only BA

$$CBQ = \sum_{qi \in BQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$$

COQ = total number of accesses to attributes by applications that access both TA and BA

$$COQ = \sum_{qi \in OQ} \sum_{\forall Sj} ref_j(q_i).acc_j(q_i)$$

Then find the point along the diagonal line of CA matrix that maximizes:

$$CTQ * CBQ - COQ^2$$

VF – Example

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=

$$\begin{array}{c|cccc} & A_1 & A_2 & A_3 & A_4 \\ \hline q_1 & 1 & 0 & 1 & 0 \\ q_2 & 0 & 1 & 1 & 0 \\ q_3 & 0 & 1 & 0 & 1 \\ q_4 & 0 & 0 & 1 & 1 \end{array}$$

Case 1:

$TA = \{ A_1 \}$, $TQ = \{ \}$,

$BA = \{A_3, A_2, A_4\}$, $BQ = \{ q_2, q_3, q_4 \}$,

$OQ = \{q_1\}$

$CTQ = 0;$

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

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

$Z = CTQ * CBQ - COQ^2 = -2025$

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

VF - Example

Case 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

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

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

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

$OQ = \{q_2, q_4\}$

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

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

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

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

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

VF - Example

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

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

VF - Example

- Case 1: $Z = -2025$
- Case 2: $Z = 3311$
- Case 3: $Z = -6084$
- Case 2 has the max value of Z
- Relation PROJ is fragmented in 2 parts:

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

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

Hoa Dinh Nguyen, PhD.
Department of Information Technology

VF - Example

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

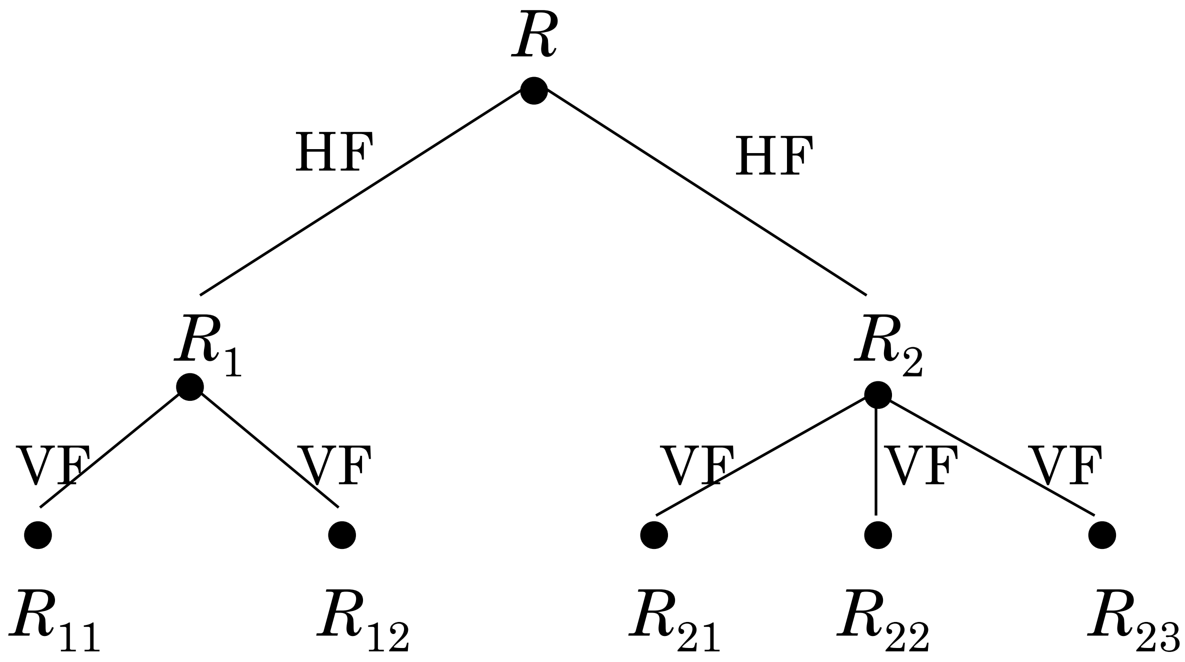
VF - Correctness

- A relation R, defined over attribute set A and key K, generates the vertical partitioning $F_R = \{R_1, R_2, \ldots, R_r\}$.
- Completeness
  - The following should be true for A:
    - $A = \bigcup A_{R_i}$
- Reconstruction
  - Reconstruction can be achieved by

$$R = R_1 \bowtie \cdots \bowtie R_n$$

- Disjointness
  - All attributes must be separated in VF
  - Tuples' ID are not considered to be overlapping since they are maintained by the system
  - Duplicated keys are not considered to be overlapping

Mixed Fragmentation



Replication and Allocation (1)

- **Replication**: Which fragments will be stored in many sites
  - Fully replication
  - Selective replication
- **Allocation**: which fragment will go to which site?

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Replication and Allocation (2)

- Replication option comparision

	Full-replication	Partial-replication	Partitioning
QUERY PROCESSING	Easy	Same Difficulty	
DIRECTORY MANAGEMENT	Easy or Non-existant	Same Difficulty	
CONCURRENCY CONTROL	Moderate	Difficult	Easy
RELIABILITY	Very high	High	Low
REALITY	Possible application	Realistic	Possible application

Allocation Problem

- Problem statement
  - Given
    - $F = \{F_1, F_2, \ldots, F_n\}$ fragments
    - $S = \{S_1, S_2, \ldots, S_m\}$ network sites
    - $Q = \{q_1, q_2, \ldots, q_q\}$ applications
  - Find the "optimal" distribution of F to S.
- Optimality
  - Minimal cost
    - Communication + storage + processing (read & update)
    - Cost in terms of time (usually)
  - Performance
    - Response time and/or throughput
  - Constraints
    - Per site constraints (storage & processing)

Allocation – Information Requests

- Database information
  - selectivity of fragments
  - size of a fragment
- Application information
  - RR_{ij}: number of reading accesses from application q_i to fragment F_j
  - UR_{ij}: number of update accesses from application q_i to fragment F_j
  - u_{ij} a matrix component showing which query updates which fragment
  - r_{ij} a matrix component showing which query reads which fragment
- Location information
  - USC_k unit cost of storing data at a site S_k
  - LPC_k unit cost of processing at a site S_k
- Network information
  - Communication cost per frame between two positions
  - Frame size

Allocation Model

General form

min(Total cost)

Subject to

response time constraint

storage constraint

processing constraint

Decision variables

$$x_{ij} = \begin{cases} 1 \text{ if fragment } F_i \text{ is stored at site } S_j \\ 0 \text{ otherwise} \end{cases}$$

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Fragment Allocation (1)

- The total cost function has two components: storage and query processing

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

- Storage cost of fragment F_j at site S_k

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

where USC_k is the unit storage cost at site S_k

- Query processing cost of a query q_i is composed of two components: processing cost PC and transmission cost TC

$$QPC_i = PC_i + TC_i$$

www.ptit.edu.vn

Fragment Allocation (2)

- Processing cost is a sum 3 components
  - Access cost (AC), integrity enforcement cost (IE), concurrency control cost CC

$$PC_i = AC_i + IE_i + CC_i$$

  - Access cost

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

Where LPC_k: the unit process cost at site k

  - Integrity and concurrency costs: Can be similarly computed, though depends on the specific constraints

Fragment Allocation (3)

- **Communication cost** is composed of two components: update processing cost and query processing cost

$$TC_i = TCU_i + TCR_i$$

- **Update cost**

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

- **Query cost**

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

Hoa Dinh Nguyen, PhD.
Department of Information Technology

Fragment Allocation (4)

- Constraint modeling
  - Constraint on responding time of query q_i
  - Execution time of q_i <= maximum allowed responding time of q_i
  - Storage constraints of site S_k

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

  - Processing constraints of site S_k

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

Fragment Allocation (5)

Solution methods

- The complexity of this allocation model/problem is NP-complete
- Correspondence between the allocation problem and similar problems in other areas
- Plant location problem in operations research
- Knapsack problem
- Network flow problem
- Hence, solutions from these areas can be re-used
- Use different heuristics to reduce the search space
  - Assume that all candidate partitioning have been determined together with their associated costs and benefits in terms of query processing
  - The problem is then reduced to find the optimal partitioning and placement for each relation
  - Ignore replication at the first step and find an optimal non-replicated solution
  - Replication is then handled in a second step on top of the previous non-replicated solution

Summary

- Designing issues
- Designing strategies (Top-down, Bottom-up)
- Fragmentation (horizontal, vertical)
- Fragment allocation and replication