

Distributed Database Management System (DDBS)

Motivation: Data is used at multiple distributed sites
(*e.g. Branch offices*).

Communication between sites is

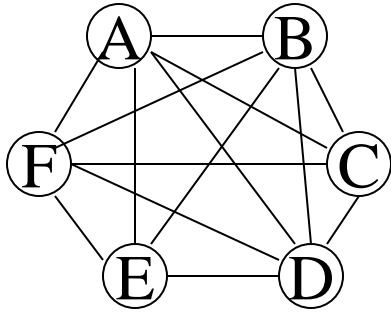
----- *costly*

----- *potentially unreliable*

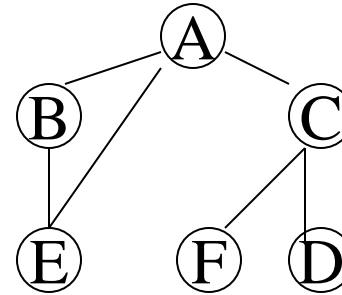
Solution:

- Allow sites to store/maintain the data they use most often/specialize in
- Sharing with other sites/HQs if combinations of data necessary

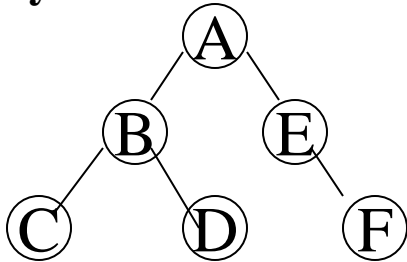
Network Topology



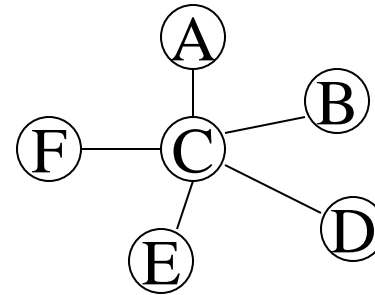
Fully connected network



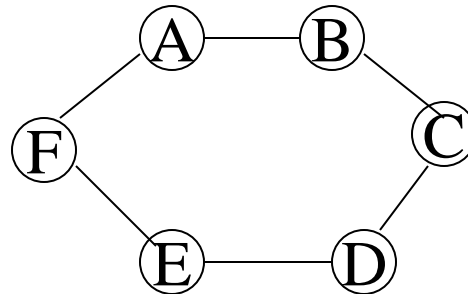
Partially connected network



Tree structured network



Star network



Ring network

**Cost/reliability
#of hops**

Tradeoffs between

1. keeping data in centralized headquarters:

- simpler maintenance
- simpler consistency enforcement
- possibly more efficient if many updates, aggregate computations


2. or distributed across branch offices:

- lower communication cost
- reliability
- parallelism *can be implemented locally*

Advantages of DDBS (heterogeneous)

- Interconnectivity of pre-existing DBs
- Expandability (don't need to replace whole system to grow)
- Cost (many small engines on PC's cheaper than mainframes) → *issue:communication costs vs. hardware computation costs.*
- Performance (place data near where used)
- Availability and reliability

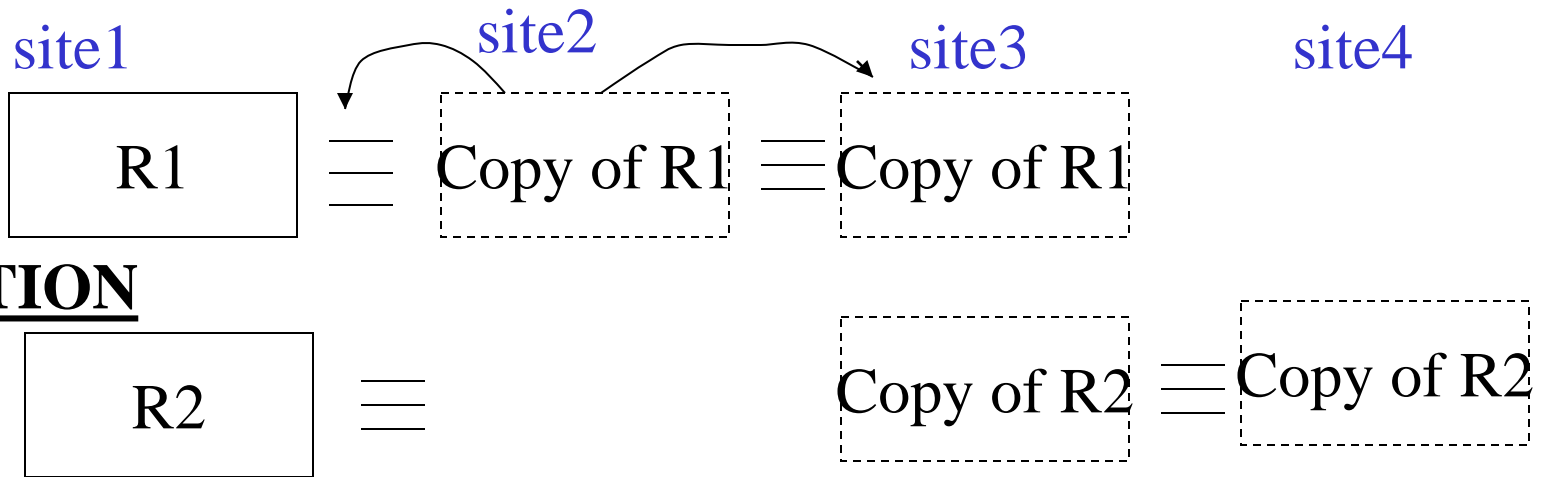
Complicating factors

- **Maintaining data consistency** (*in face of replication and sharing*)
- **Distributed directory management** (*who controls mapping of data to sites*)
- **Security**
- **Heterogeneous Databases**
 *different database architectures*

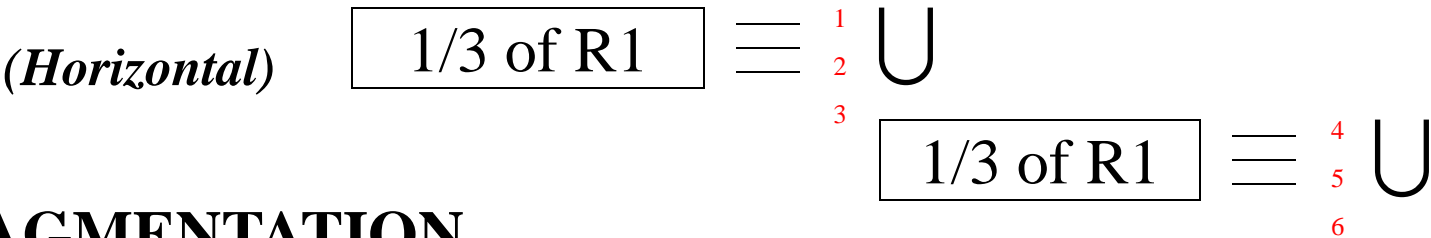
Distributed Database Design Issues

Options for storing a relation \underline{R} across multiple sites:

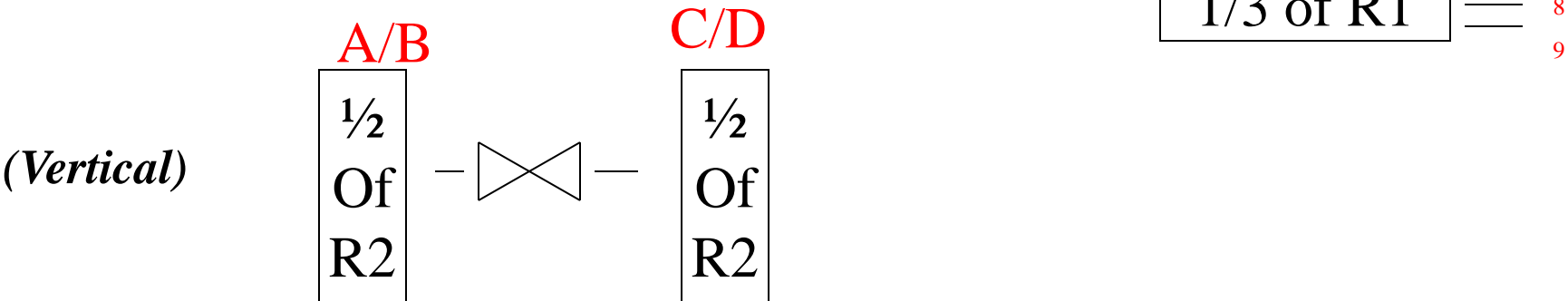
- Replication (*maintain copies/replicas of R on multiple sites*)
- Fragmentation (*Relation store in fragments/ pieces on multiple sites*)
- combination of both



REPLICATION



FRAGMENTATION



Replication

Non redundant
allocation



- Issues: (whole database replication v.s. no replication)
 - what to replicate? (*all relations or only frequently user shared data*)
 - where to replicate? (*function of communication costs, usage needs, resources*)
 - which relations to replicate?
 - "primary copy" of relation (*simplifies consistency enforcement, but where located?*)

Replication (cont)

- Advantages:

- Improved availability (*multiple sources for a relation if a site is down*)



- Increased parallelism (*sites can process (primarily) read-only operations in parallel, minimizing data transfer*)

(well suited for read-only, majority read-only data access)

Replication (cont)

- Disadvantages:

- o problems/overhead for writes/updates

- o costs of consistency enforcement

- updates propagated to all sites
(communication costs)

- costs of synchronization/locking for consistency enforcement on update greater than in single source models.

- Complicates concurrency and recover

- Replication inefficient in databases with frequent updates

FRAGMENTATION

- Vertical
- Horizontal
- mixel

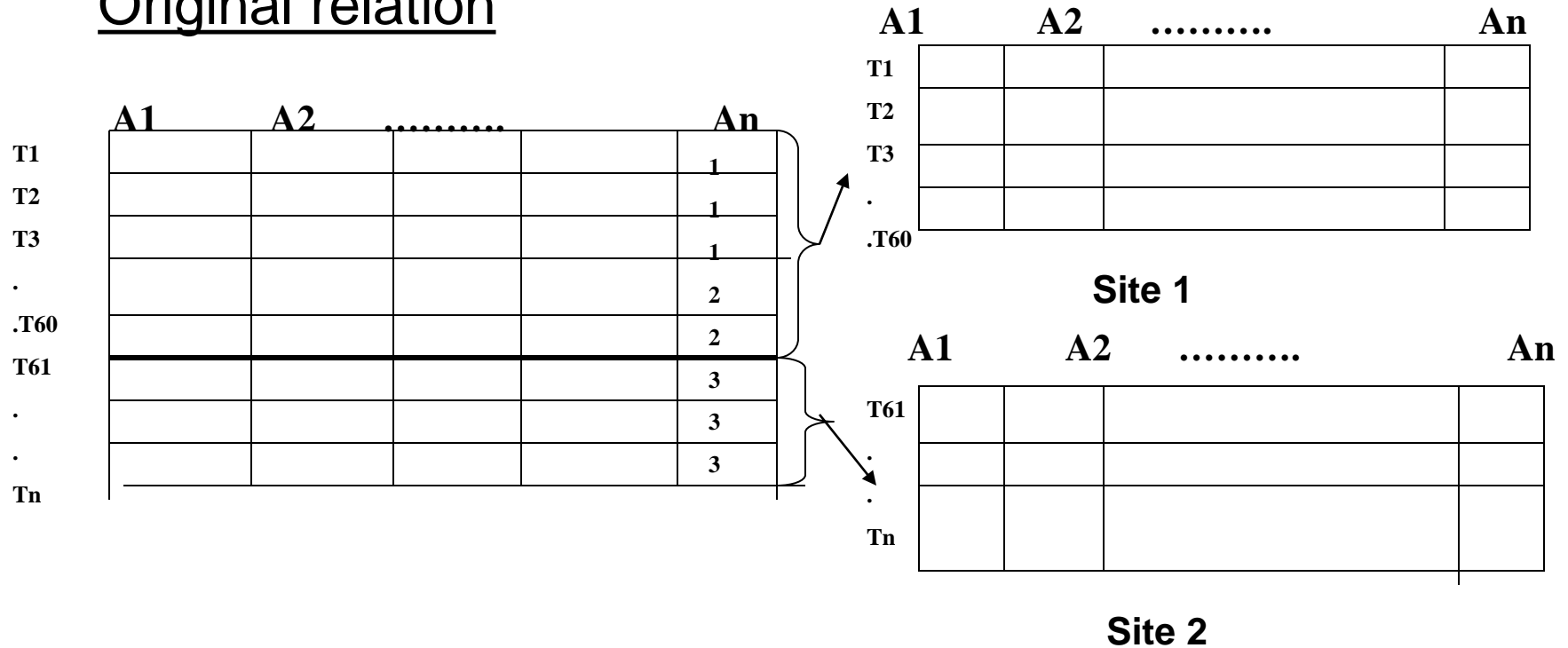


Issues:

- completeness: Every tuple/attribute in some fragment
- reconstruction: easy way of reconstructing full relation
- transparency

HORIZONTAL FRAGMENTATION

Original relation



-Fragments contain subsets of complete tuples (all attributes at all sites)

How to reconstruct

$$R = R_{s1} \cup R_{s2} \cup \dots \cup R_{sn}$$

Horizontal Fragmentation

- Example Usefulness:
 - Each branch office maintains complete attribute set of its employees
(salary,benefits,address/phone,departments,projects,etc.)
 - Site of Fragment easily determined by a key attribute value -e.g. Branch_office*

VERTICAL FRAGMENTATION

Original
Relation

(R)

	A1	A2	A3	A4
t1				
t2				
tn				

How to Reconstruct:

$$R = R_{s1} \bowtie R_{s2} \bowtie \dots \bowtie R_{sn}$$

TID – Tuple ID

Hidden Attribute to
ensure account
and simple join
reconstruction

R_{s1}

	A1	A2	TID
t1			1
t2			2
tn			n

SITE1



TID	A3	A4
1		
2		
n		

SITE2

R_{s2}

	TID
t1	
t2	
tn	



$R_{s1}.TID = R_{s2}.TID$

Join condition

VERTICAL FRAGMENTATION

Example usefulness:

Salary Office

Benefits Office

Directory (Name|address|phone|fax)

Dependents Management Office

each control their own appropriate

attribute for all corporate branch offices

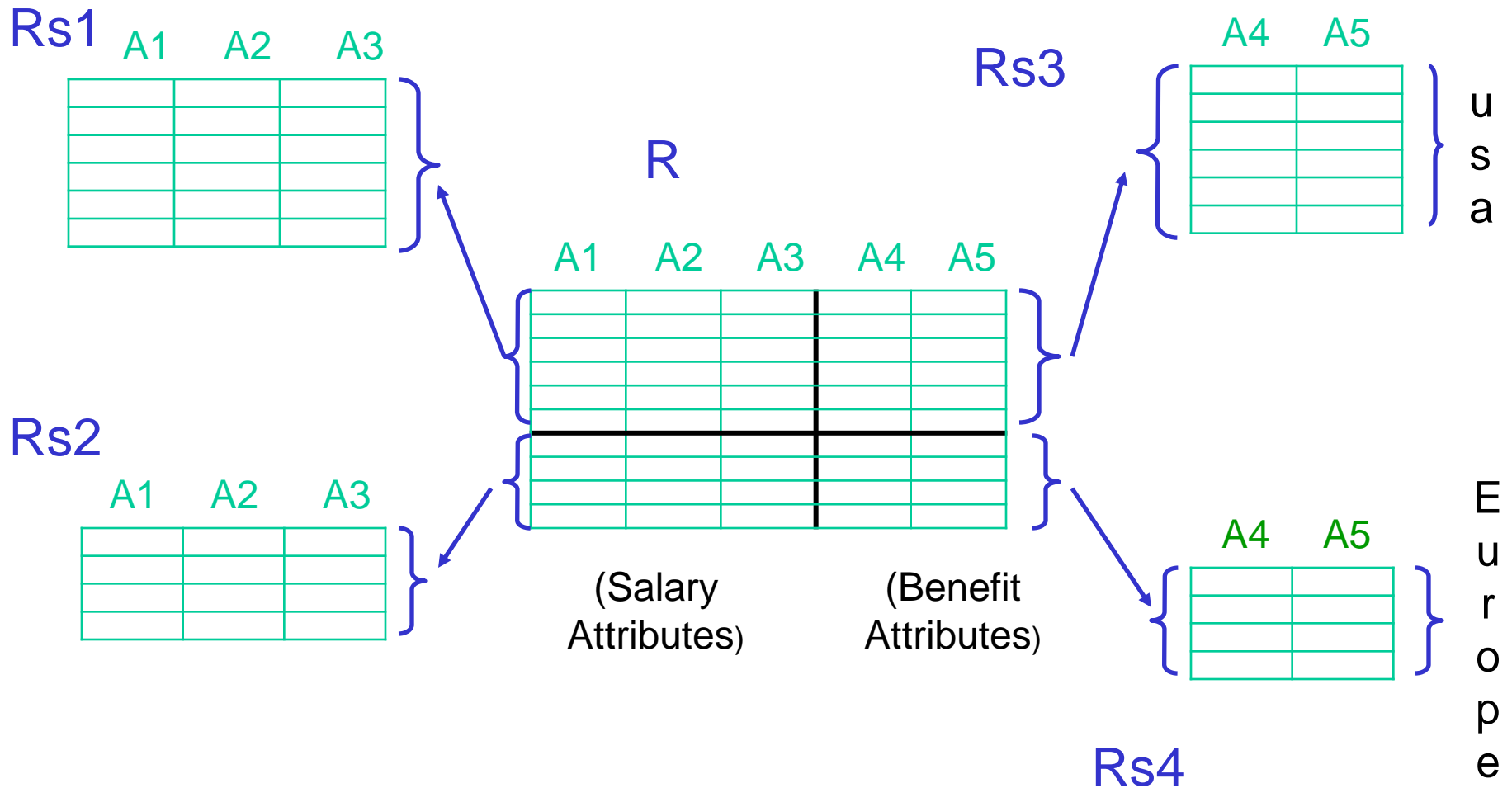
VERTICAL – Attribute-centered management

(keep all instances of an attribute in one place)

HORIZONTAL – tuple/individual-centered management

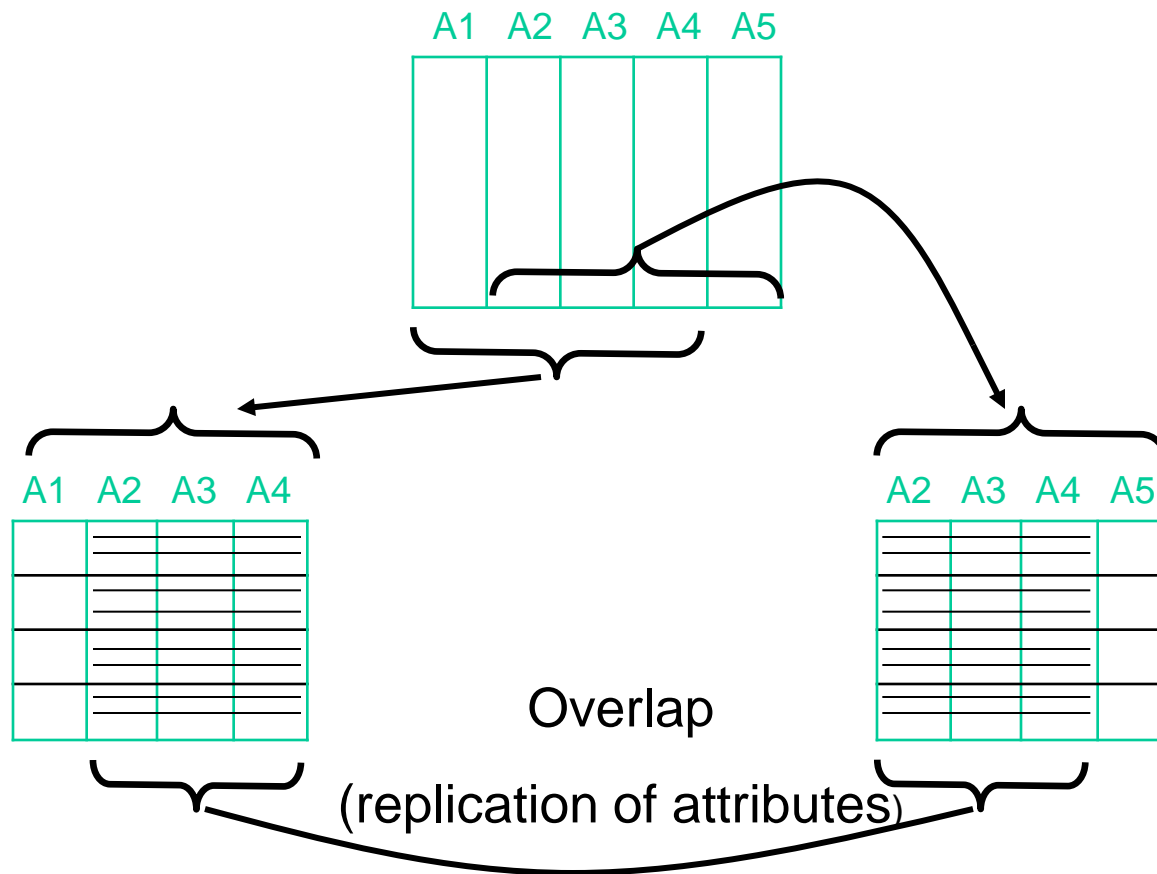
(keep all values of a tuple in one place)

MIXED FRAGMENTATION



REPLICATION and FRAGMENTATION

Partition of Attributes/tuples need not be disjoint



TRANSPARENCY

Fragmentation Transparency

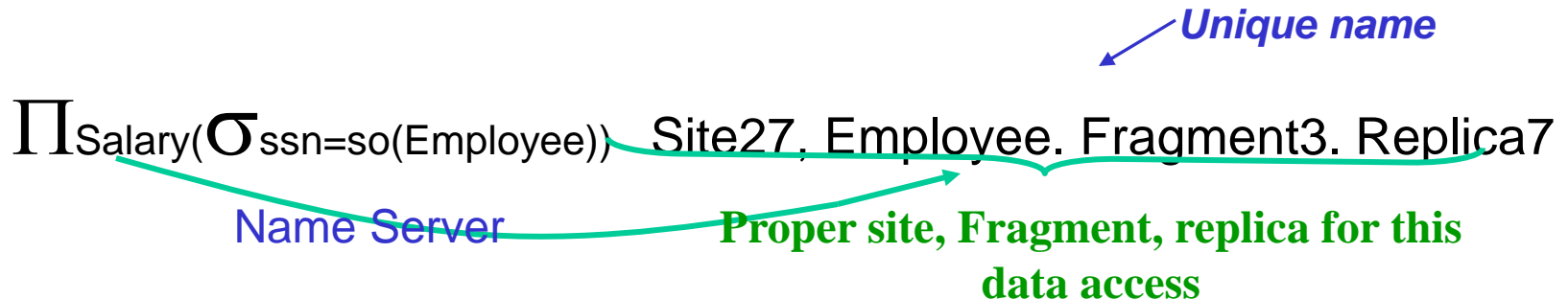
-User doesn't need to know mapping between relations and fragmented subrelations

Replication Transparency

-User doesn't need to know about existence or location of other copies (treat as if single copy of DB)

Location and Naming Transparency

-User shouldn't need to know about location and full names of data on the server

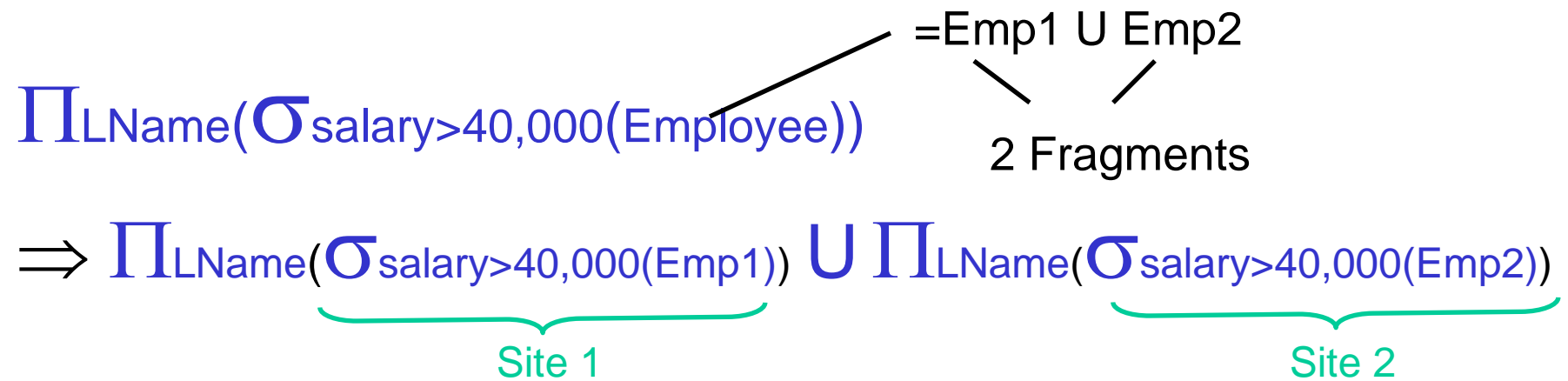


QUERY PROCESSING IN DDMS

Issues1:

Parallel Processing across Fragments

Horizontal
fragmentations





Execution in Parallel on fragments


and union results together

Site1 Site2 Site3
 50K 1K 3K

$(A \bowtie B) \bowtie C$

1K  3K
 
 0.5K

$A \bowtie (B \bowtie C)$

50K  0.5K
 
 0.5K

Joins- symmetric and
 associative

Parallel Processing

$(\sigma_{xx}(A)) \bowtie (B \bowtie C)$

QUERY PROCESSING IN DDBS

Join Strategies

$R = \prod_{Fnames, Cnames, Dnames} (Employee \bowtie Department)$

Site 3

100 records, 2000 bytes

Site 1

10,000 records,
1,000,000 bytes

Mg rsn
to ssn

Site 2

100 records, 3000
bytes

Strategies:

- 1) Ship both relations to the result site and join there
1,003,000 bytes transferred
- 2) Ship employee to 2, join at 2, results to 3
1,002,000 bytes transferred
- 3) Ship Department to 1, join at 1, results to 3
5,000 bytes transferred

⇒ minimize total communication cost of data transfer

RECOVERY IN DDBS

- transaction managers / coordinators
- log managers

Problems:

- failure of site
- failure of link
- loss of messages

} Difficult to know
which had occurred

if server is down, elect new server \Rightarrow what about network partitioning?

