Database Management Systems

- Distributed Databases

Washington University in St. Louis

What does it mean to be distributed?

- Multiple nodes connected by a network
- Data on the nodes is logically related
- The nodes do not need to be homogeneous
- The network matters a lot:
  - Topology
  - Proximity

Transparency

- Do our users need to be aware of the fact that a database is distributed?

- Data organization transparency
  - Location transparency
  - Naming transparency

- Replication transparency

- Fragmentation transparency
  - Horizontal
  - Vertical

Autonomy

- Can nodes operate independently?
  - Design autonomy
  - Communication autonomy
  - Execution autonomy

Reliability and Availability

- Reliability: probability that a system is running

- Availability: probability that the system is continuously available

- How do we construct reliable systems?
  - Fault tolerance

Advantages

- Flexibility for large operations

- Increased reliability and availability

- Improved performance
  - Data localization

- Easier Expansion

Additional Functions

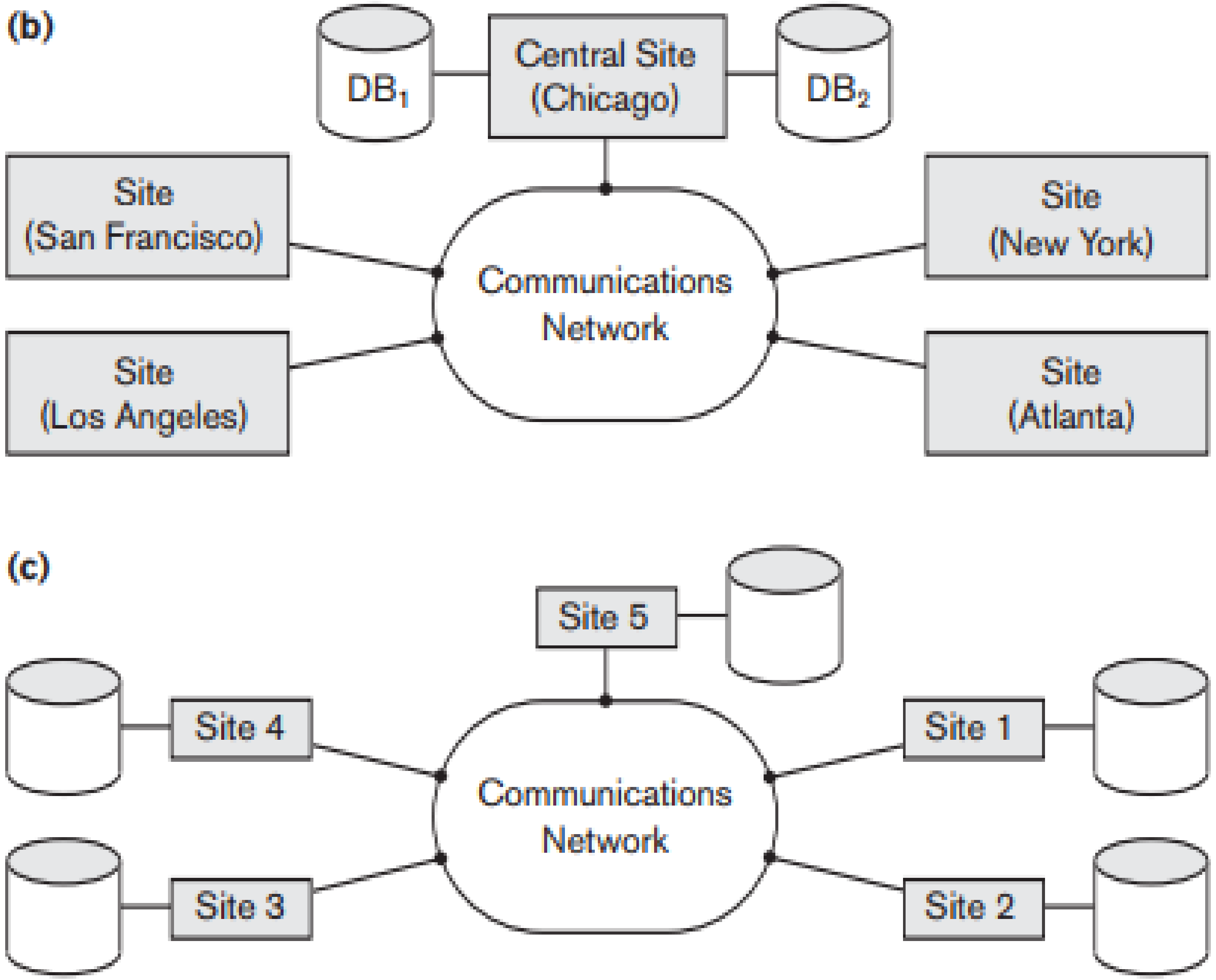
- Manage data distribution
- Distributed query processing
- Replication management
- Distributed transaction management
- Recovery
- Security

Distributed Database Properties

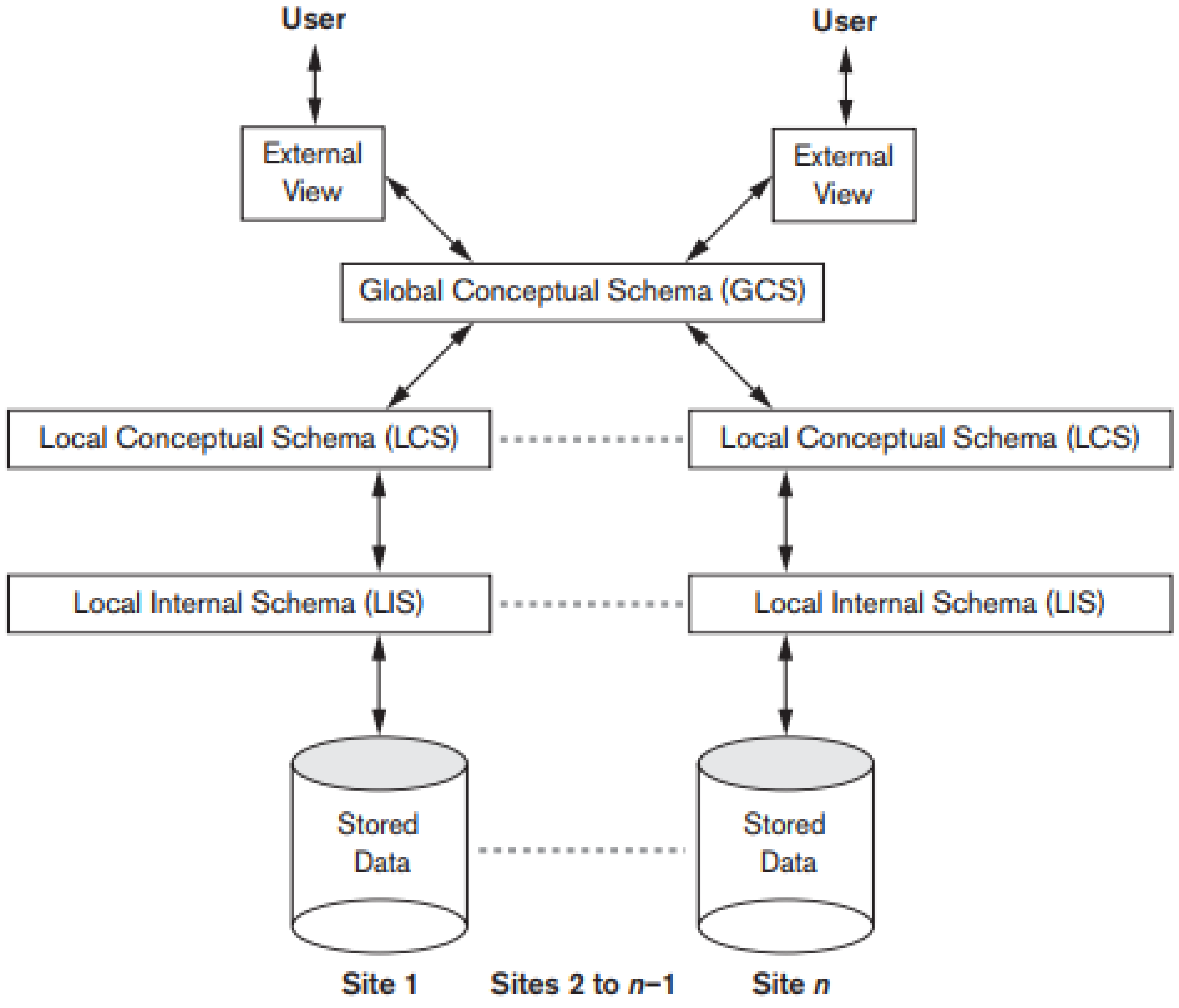
- Degree of homogeneity

- Degree of local autonomy

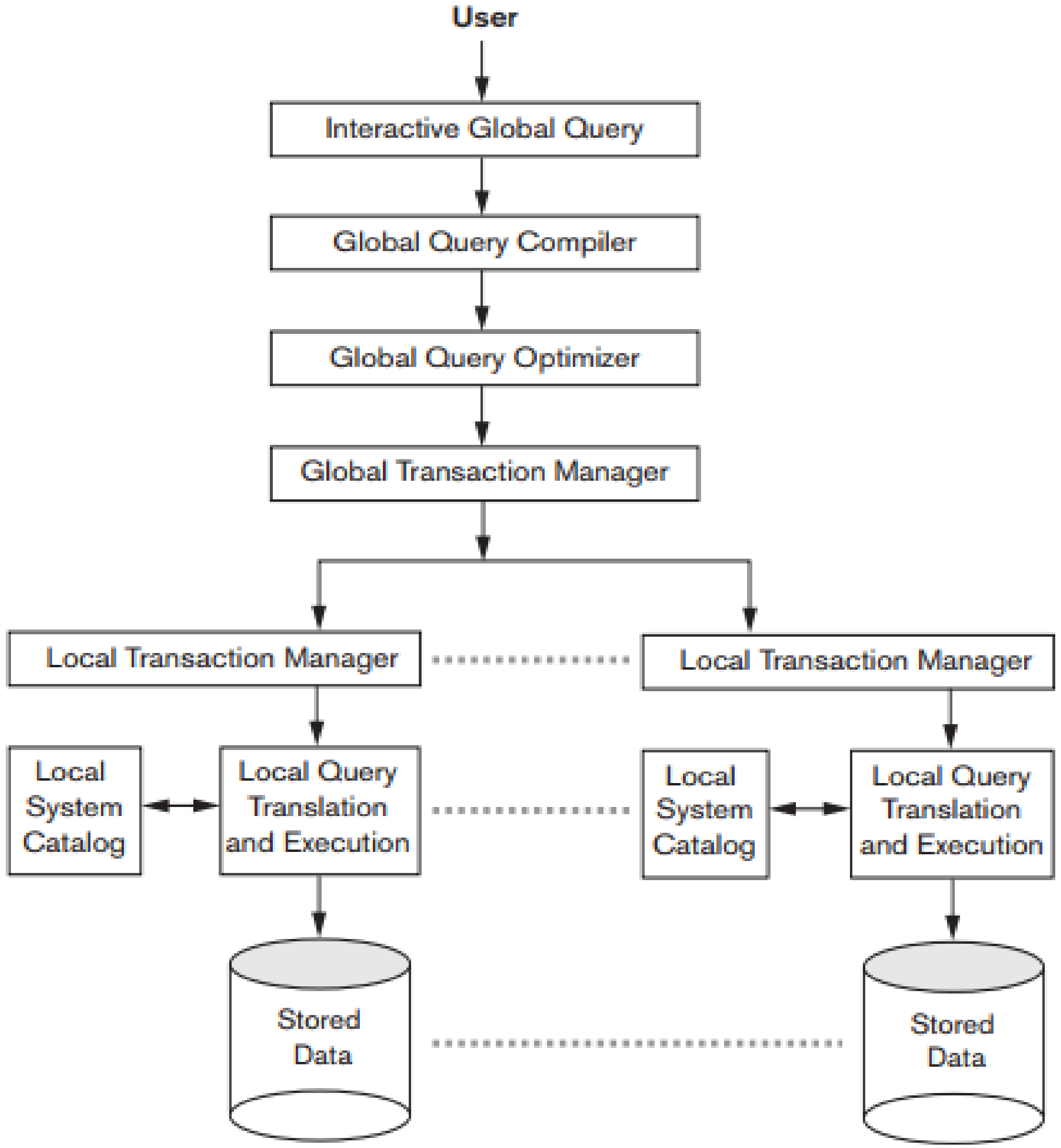
Architecture



Schema Architecture


User
User
External View
External View
Global Conceptual Schema (GCS)
Local Conceptual Schema (LCS)
Local Conceptual Schema (LCS)
Local Internal Schema (LIS)
Local Internal Schema (LIS)
Stored Data
Stored Data
Site 1
Sites 2 to *n*–1
Site *n*

Component Architecture


User
Interactive Global Query
Global Query Compiler
Global Query Optimizer
Global Transaction Manager
Local Transaction Manager
Local Transaction Manager
Local System Catalog
Local Query Translation and Execution
Local System Catalog
Local Query Translation and Execution
Stored Data
Stored Data

Data Fragmentation

- Where should the data go?

- Break the data down into logical units
  - What pieces of our database make sense as logical units?

Types of Fragmentation

- Horizontal Fragmentation
  - Break a relation down into subsets of tuples
    - Specify a set of conditions for this purpose
    - What about relationships?

- Vertical Fragmentation
  - Break a relation down into subsets of columns
  - How do we reconstruct the original tuples?

- We can use relational operations to specify fragmentation
  - Which ones?
  - Which operations will reconstruct fragmented tuples?

Replication and Allocation

- Replication improves availability
  - How?

- Should we replicate everything?
  - Pros?
  - Cons?

Example

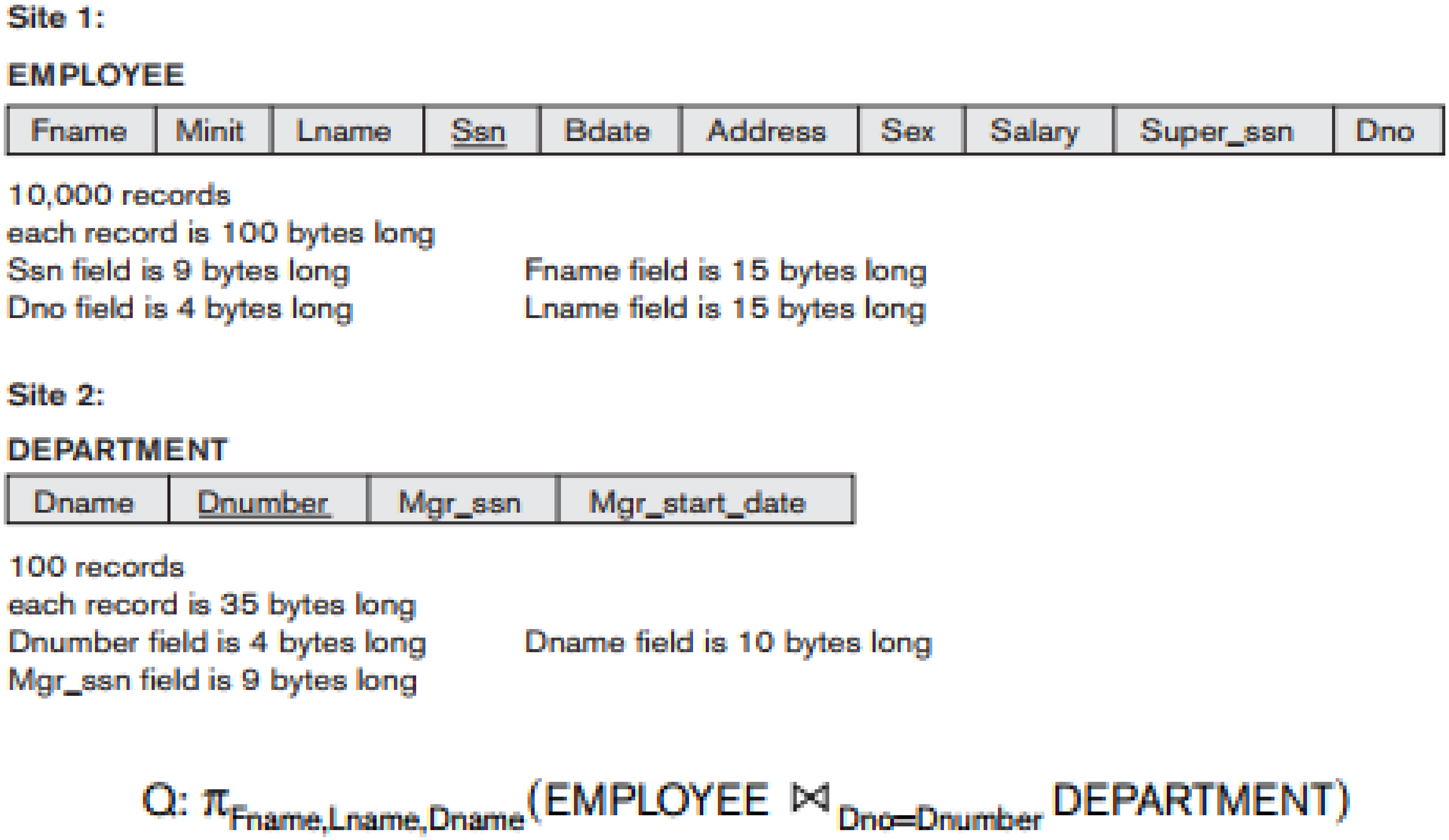
- Three sites
  - Site 1 is HQ, accesses all data
  - Site 2 is the home of department 5
    - Needs access to employee and project info
  - Site 3 is the home of department 4
    - Needs access to employee and project info

- Which part of the DB should be at each site?
  - What kind of fragmentation is required?
  - What should we do if an employee from department 5 works on a project owned by department 4?

Query Processing

- Processed in stages
  - Mapping
  - Localization
  - Global Optimization
    - What are we optimizing here?
  - Local Optimization

Data Transfer Costs



Site 1:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Dno field is 4 bytes long

Fname field is 15 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Mgr_ssn field is 9 bytes long

Dname field is 10 bytes long

Q: $\pi_{\text{Fname,Lname,Dname}}(\text{EMPLOYEE} \bowtie_{\text{Dno=Dnumber}} \text{DEPARTMENT})$

Data Costs

- We submit our query at a third site
  - Assume each record in the result is 40 bytes long

- Three options
  - Transfer both relations to third site, perform the join
  - Transfer EMPLOYEE to site 2, perform the join there, send the results
  - Transfer department to site 1, perform the join there, send the results

- Which option results in the least amount of data being sent?

Practice Problem

- Redo the following example using the following query:

$$Q': \pi_{\text{Fname,Lname,Dname}}(\text{ DEPARTMENT } \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$

- In the previous examples we assumed that the queries were being submitted from a third site. What if the query is being submitted from site 2? Compute the data costs.

Decomposition

- If multiple copies exist, which one should we read from?
  - What if we need to update that information?

- Where do we go to find out how many copies of a piece of information there are and where they are stored?

Transaction Management

- Since data will be accessed from many sites, we need a way to coordinate
  - Global transaction manager is in charge of each transaction

- Idea: Two phase commit
  - Some problems though…

Three Phase Commit

- Break the commit into two phases:
  - Prepare to commit
  - Commit

- Prepare-to-commit
  - All participants vote on whether the commit should happen
  - If yes, then proceed to commit as normal
  - What do we do on a crash?

Concurrency Control

- Problems:
  - Multiple copies of the same data
  - Failure and recovery of individual sites
  - Failure of communication
  - Distributed Commits
  - Distributed Deadlock

Distinguished Copy

- How do we track locks for distributed items?
  - Idea: make a distinguished copy
  - All locking requests are sent to this copy

- Where does this distinguished copy live?
  - Called the coordinator site

Coordinator Sites

- Primary Site
  - One site has all the distinguished copies
  - Disadvantages?

- Primary with backup

- Primary Copy

- What happens when a site goes down?

Voting Method

- No distinguished copy
  - Lock requests are sent to all sites with the item
  - Must receive a majority of locks
  - Advantages?
  - Disadvantages?

Distributed Catalogs

- How do we track catalog information with multiple sites?
  - One copy?
  - Many copies?
- Centralized vs. Replicated

Review Questions

- Name one advantage and one disadvantage of replication.

- Under what circumstances would it be preferable to use a distributed catalog as opposed to a centralized catalog.

- What is the difference between reliability and availability? How does distribution affect these properties?

Review Questions

- When replication is used, the database must decide which replica to use for a given query. What factors will the database use to make this decision?

- Assuming that we are not using any replication, which partitioning scheme will take up more space? Vertical or horizontal? Why?

Practice Problem

A company has an existing database that it uses to track product shipments. We wish to distribute the database in the following way:

- There are two shipping centers. Shipping center one is responsible for all video game shipments, while shipping center two is responsible for all computer shipments. We wish to place the data for these types of products (and only these types of products) at each site. We wish to keep the information for all products at a third site, the company headquarters.
- Each product has information about how much our company paid for it as well as how much we wish to sell it for. This pricing information is needed at the company headquarters, but is not needed at the shipping centers.
- We also must keep track of what vendor each product was purchased from. This vendor information is kept only at the shipping centers, not at company headquarters.

What kind of fragmentation is being used?

Practice Problem

The following query is submitted to the fragmented database at company headquarters. We wish to compute the results of this query and then make sure that the results end up back at company headquarters.

$$\pi_{VendorName,ProductName,Price}(VENDOR \bowtie_{vendor.vendorid=products.vendorid} PRODUCTS)$$

You are given the following column sizes:

VendorName (vendors table): 10 bytes

ProductName (products table): 10 bytes

Price (products table): 4 bytes

VendorID (vendors table): 4 bytes

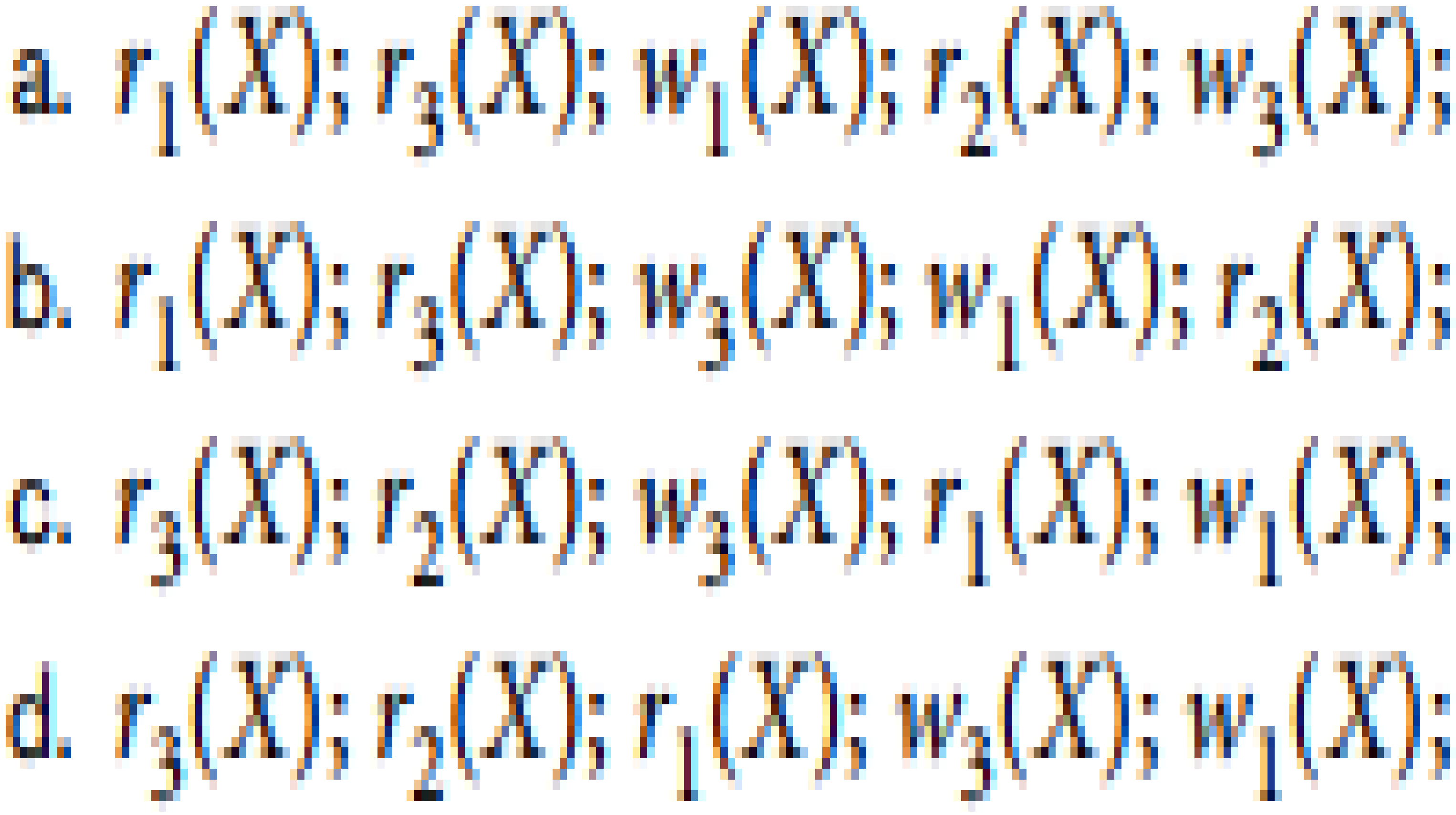
VendorID (products table): 4 bytes

The Products table at shipping center one contains 100 rows, the Products table at shipping center 2 contains 150 rows, and the Vendors table contains 50 rows.

What is the minimum number of bytes that must be sent to complete this query?

Practice Problem

- Which of the following schedules is conflict serializable?
  - Determine an equivalent serial schedule



a. $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$

b. $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$

c. $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$

d. $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X);$

Practice Problem

- Sketch the conflict graph for each schedule below
  - Come up with an equivalent serial schedule, if possible.

$$T_1: r_1(X); r_1(Z); w_1(X);$$

$$T_2: r_2(Z); r_2(Y); w_2(Z); w_2(Y);$$

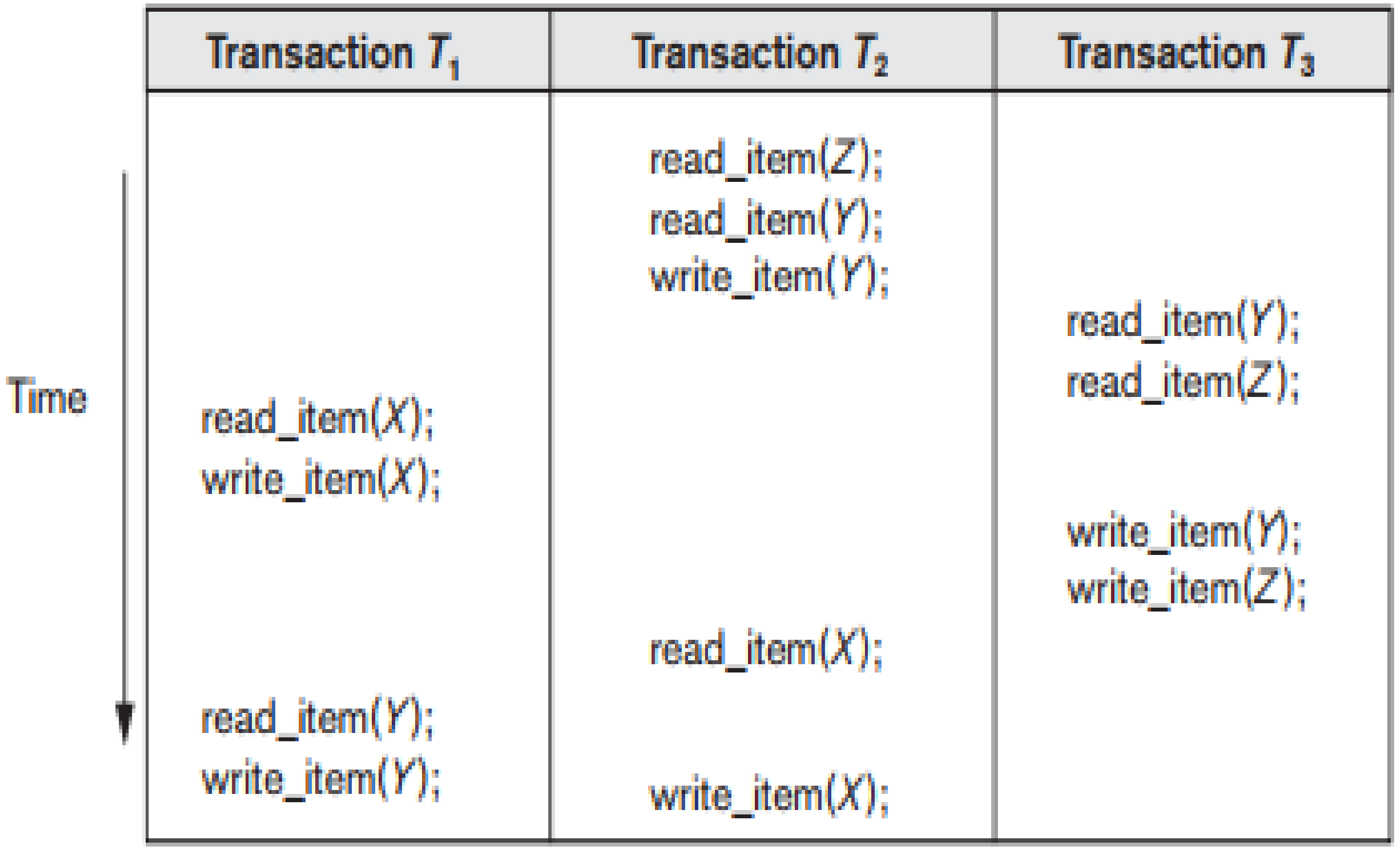
$$T_3: r_3(X); r_3(Y); w_3(Y);$$

$$S_1: r_1(X); r_2(Z); r_1(Z); r_3(X); r_3(Y); w_1(X); w_3(Y); r_2(Y); w_2(Z); w_2(Y);$$

$$S_2: r_1(X); r_2(Z); r_3(X); r_1(Z); r_2(Y); r_3(Y); w_1(X); w_2(Z); w_3(Y); w_2(Y);$$

Practice Problem

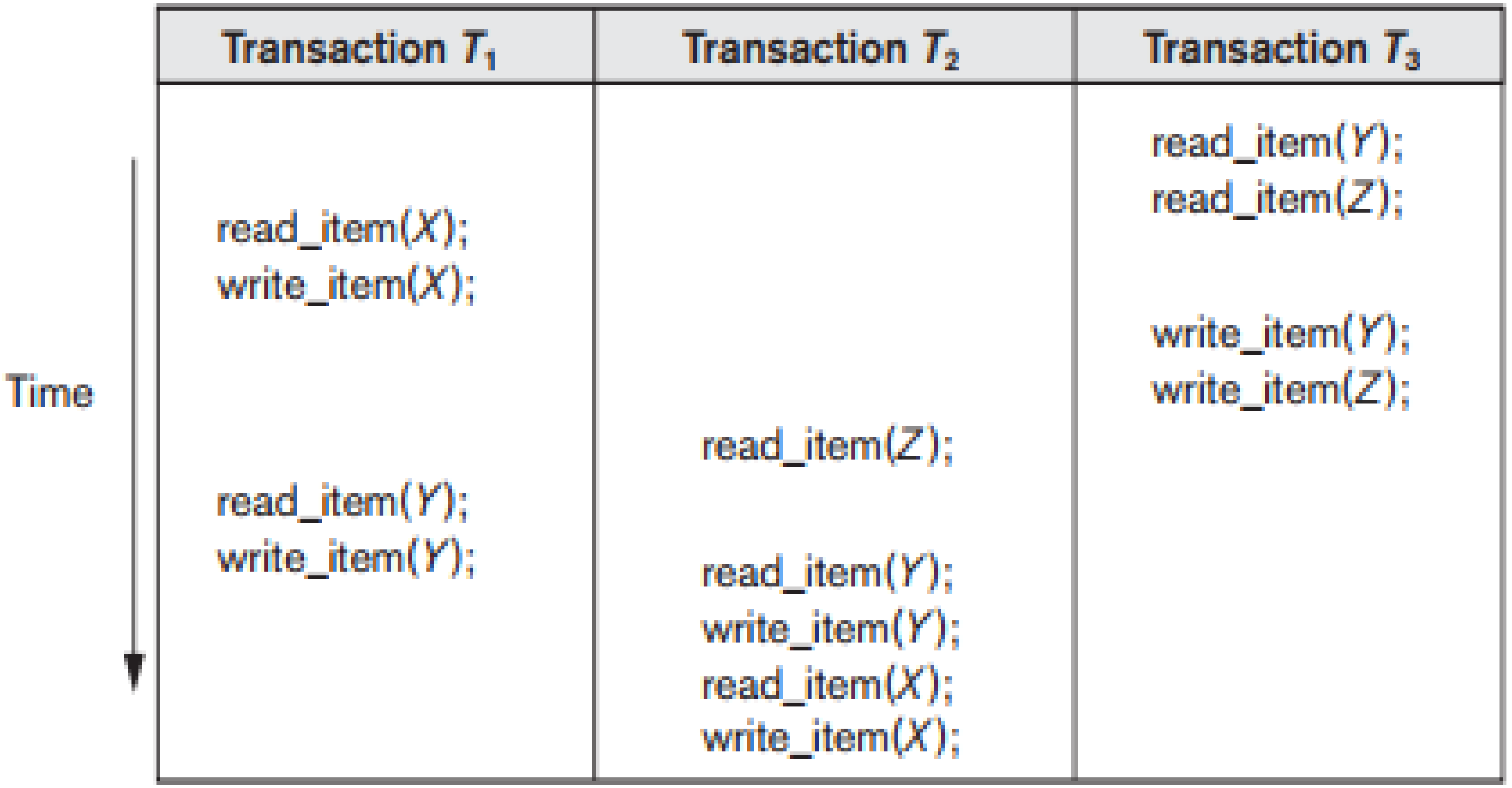
- Insert locks into the following schedule such that it satisfies strict two-phase locking. Will your schedule suffer from deadlock?
- Is this schedule serializable?



Transaction T_1	Transaction T_2	Transaction T_3
	read_item(Z);	
	read_item(Y);	
	write_item(Y);	
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(X);	
read_item(Y);		
write_item(Y);	write_item(X);	

Practice Problem

- Insert locks into the following schedule such that it satisfies strict two-phase locking. Will your schedule suffer from deadlock?
- Is this schedule serializable?



	Transaction T_1	Transaction T_2	Transaction T_3

			read_item(Y);
			read_item(Z);

	read_item(X);
	write_item(X);		

Time			write_item(Y);
			write_item(Z);

		read_item(Z);	

	read_item(Y);
	write_item(Y);		

		read_item(Y);
		write_item(Y);
		read_item(X);
		write_item(X);	

Schedule F

- Consider the following relations:

BOOKS(Book#, Primary_author, Topic, Total_stock, $price)
BOOKSTORE(Store#, City, State, Zip,
Inventory_value)

- Consider that BOOKS are fragmented by $price amounts into:
  - B1: BOOK1: $price up to $20
  - B2: BOOK2: $price from $20.01 to $50
  - B3: BOOK3: $price from $50.01 to $100
  - B4: BOOK4: $price $100.01 and above
- Similarly, BOOK_STORES are divided by zip codes into:
  - S1: EAST: Zip up to 35000
  - S2: MIDDLE: Zip 35001 to 70000
  - S3: WEST: Zip 70001 to 99999

- What kind of fragmentation exists in this database?

- Consider the query:

```
SELECT Book#, Total_stock
FROM Books
WHERE $price > 15 AND $price < 55;
```

- Assume that fragments of BOOKSTORE are nonreplicated and assigned based on region. Assume further that BOOKS are allocated as:

  EAST: B1, B4
  MIDDLE: B1, B2
  WEST: B1, B2, B3, B4

- Assuming the query was submitted in EAST, what remote queries does it generate?