Database Management Systems

- Distributed Databases

Doug Shook

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

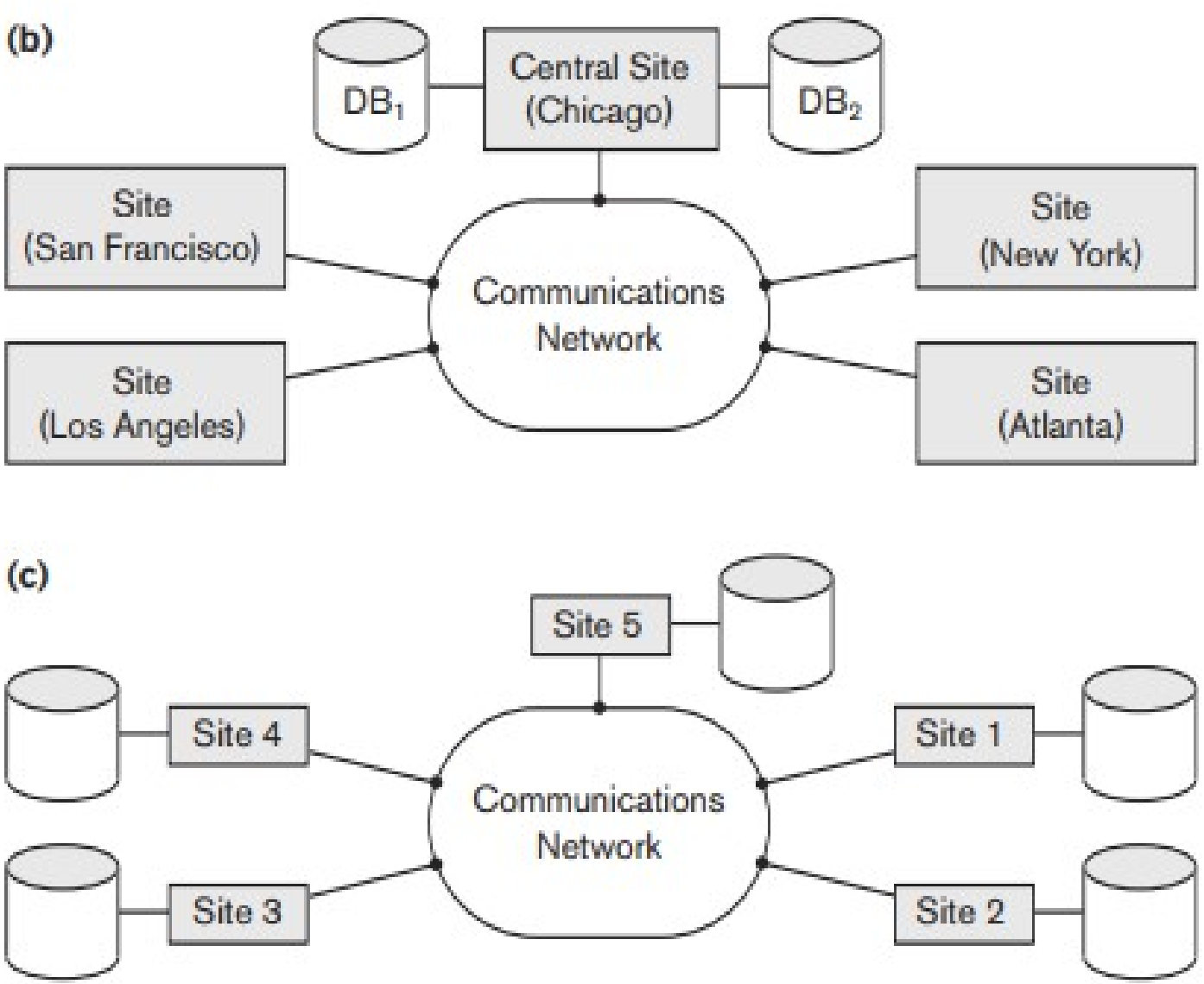
Additional Functions

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

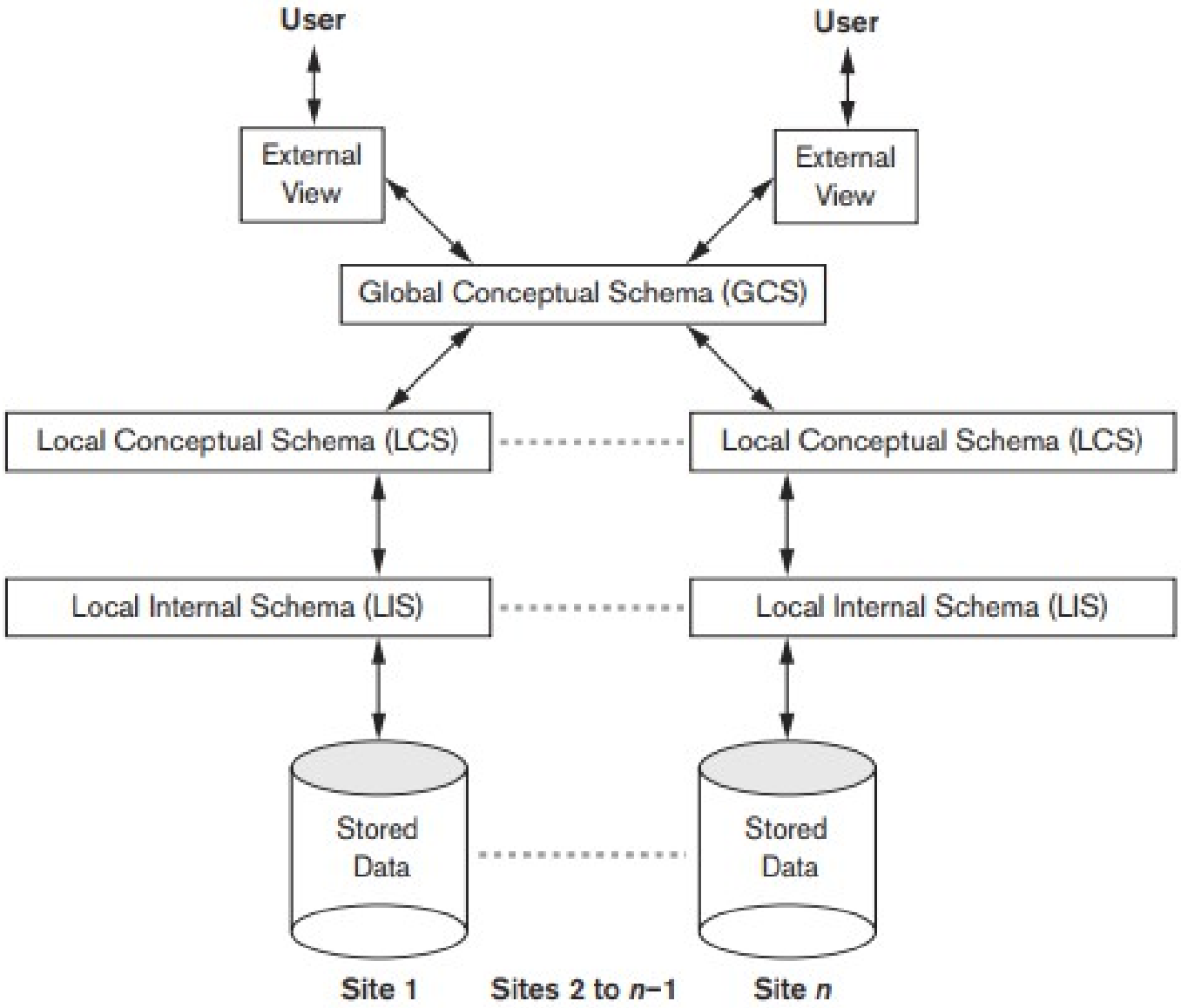
Distributed Database Properties

- Degree of homogeneity
- Degree of local autonomy

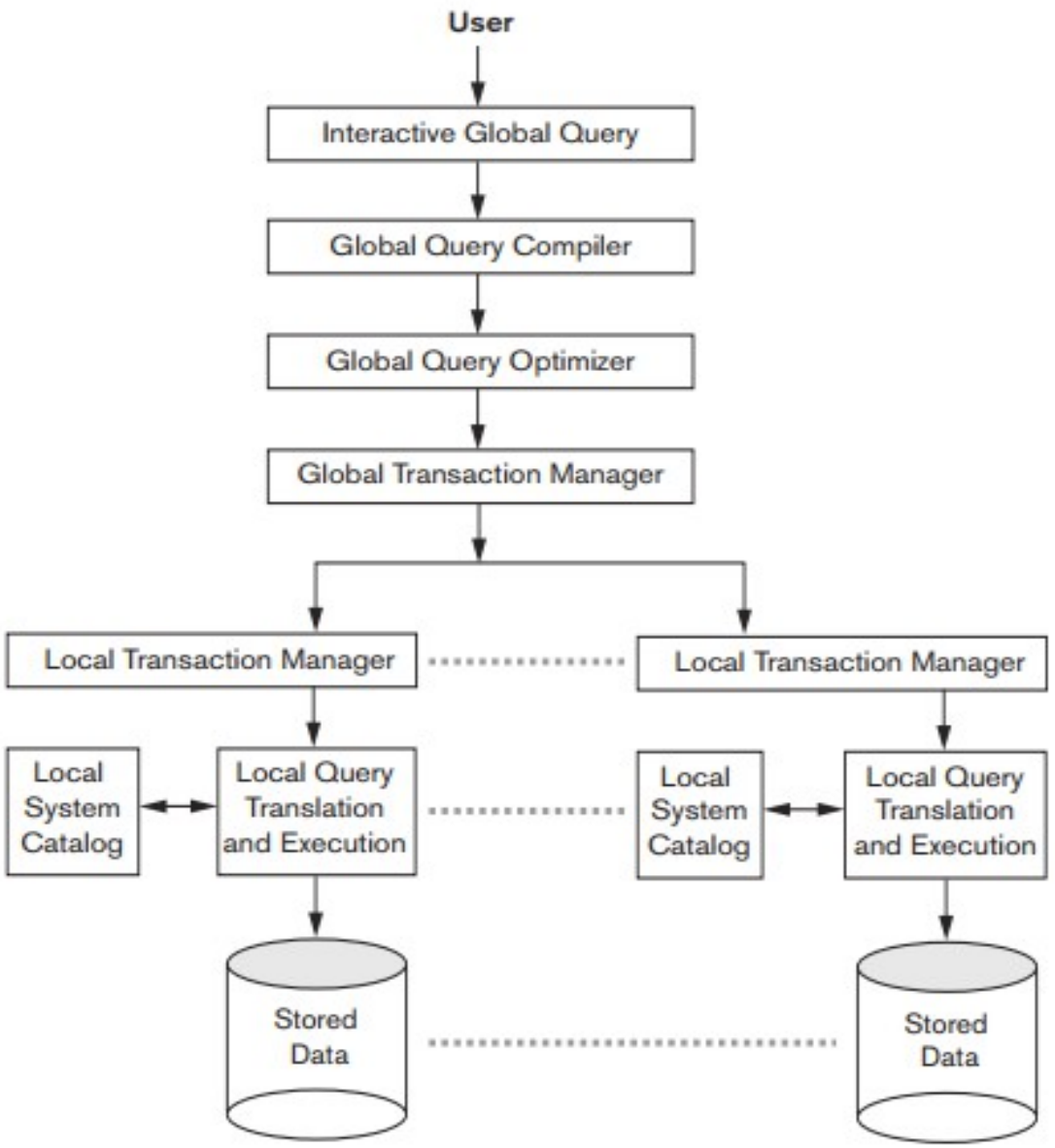
Architecture



Schema Architecture



Component Architecture



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
Fname field is 15 bytes long
Dno field is 4 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
Dname field is 10 bytes long
Mgr_ssn field is 9 bytes long

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

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:

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

- 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
- Unfortunately, two phase commits have some problems in this setup...
  - Like what?

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

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

- For the following transactions, determine what locks would need to be acquired, then come up with a schedule that would satisfy two phase locking.

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

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

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

Practice Problem

- Could the following schedule suffer from deadlock? Explain how you know. If deadlock is a problem, explain how it could be resolved.

Time

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

- Could the following schedule suffer from deadlock? Explain how you know. If deadlock is a problem, explain how it could be resolved.

Time

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

Practice Problem

- We wish to distribute our course tracking database so that each department is considered a separate site.
  - Explain what kinds of partitioning would need to take place to accomplish this.
  - Consider a query that reports all of the courses that a student has taken across all departments
    - What should the data transmission pattern be?
    - Will distribution improve or degrade the performance of this query?