Bharathidasan University

Centre for Differently Abled Persons

Khajamalai Campus, Tiruchirappalli
Tamil Nadu, India

Bachelor of Computer Applications

(For Students with Speech and Hearing Impairment)

Course: Database Management System

Unit - 4

Compiled By
Dr.M.Prabavathy
(Assistant Professor)
Dr.R.Nandhakumar

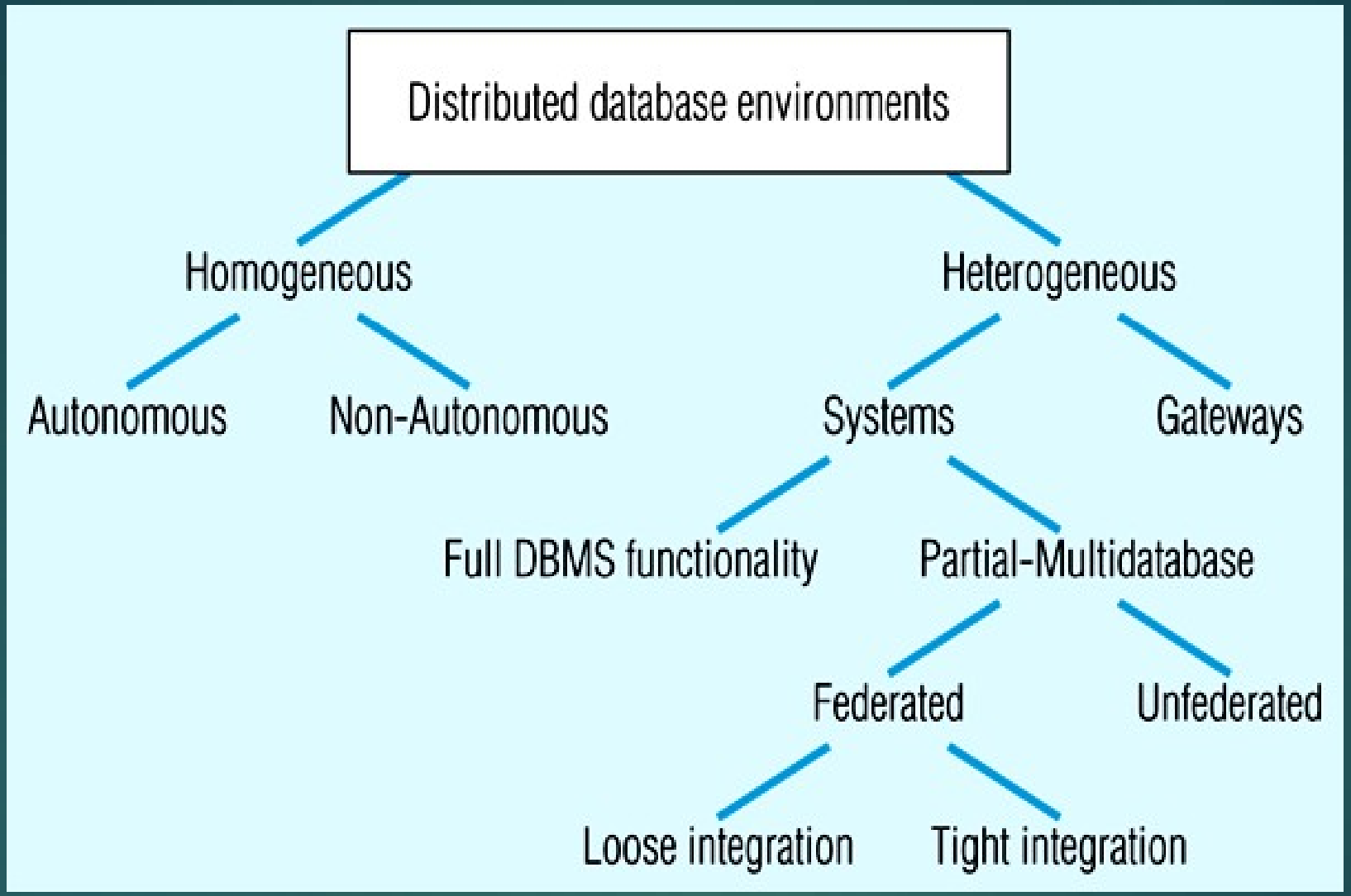
Distributed Database

- A *single logical database* that is spread physically across computers in multiple locations that are connected by a data communications link

- **Decentralized Database:** A collection of *independent* databases on non-networked computers

Reasons for Distributed Database

- Business unit autonomy and distribution
- Data sharing
- Data communication costs
- Data communication reliability and costs
- Multiple application vendors
- Database recovery
- Transaction and analytic processing



Distributed database environments (adapted from Bell and Grimson, 1992)

Distributed Database Options

- Homogeneous - Same DBMS at each node
  - Autonomous - Independent DBMSs
  - Non-autonomous - Central, coordinating DBMS
  - Easy to manage, difficult to enforce
- Heterogeneous - Different DBMSs at different nodes
  - Systems – With full or partial DBMS functionality
  - Gateways - Simple paths are created to other databases without the benefits of one logical database
  - Difficult to manage, preferred by independent organizations

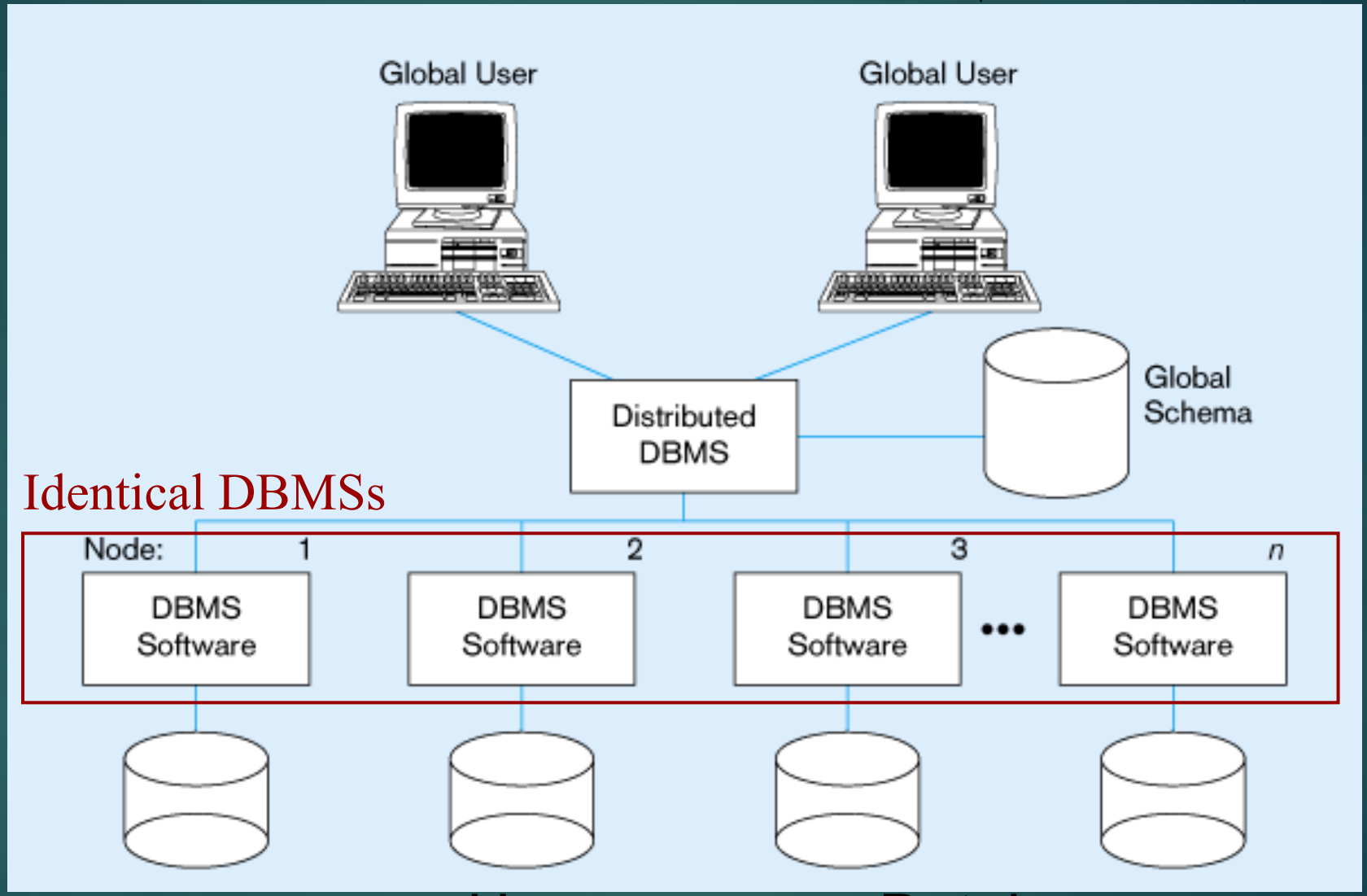
Distributed Database Options (cont.)

- **Systems** - Supports some or all functionality of one logical database
  - Full DBMS Functionality - All distributed DB functions
  - Partial-Multi database - Some distributed DB functions
    - Federated - Supports local databases for unique data requests
      - Loose Integration - Local dbs have their own schemas
      - Tight Integration - Local dbs use common schema
    - Unfederated - Requires all access to go through a central, coordinating module

Homogeneous, Non-Autonomous Database

- Data is distributed across all the nodes
- Same DBMS at each node
- All data is managed by the distributed DBMS (no exclusively local data)
- All access is through one, global schema
- The global schema is the *union* of all the local schema

Source: adapted from Bell and Grimson, 1992.

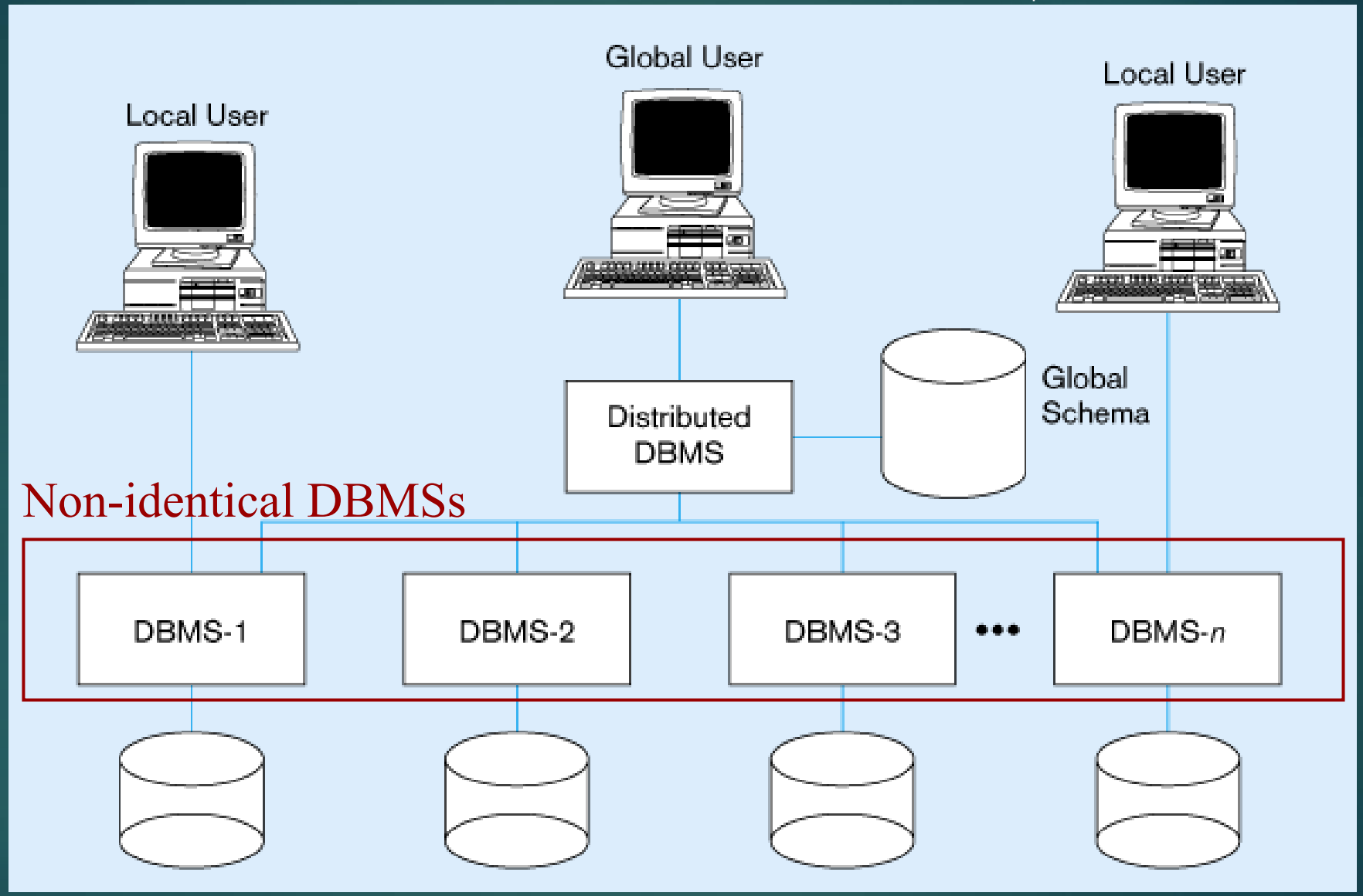


Homogeneous Database

Typical Heterogeneous Environment

- Data distributed across all the nodes
- Different DBMSs may be used at each node
- Local access is done using the local DBMS and schema
- Remote access is done using the global schema

Source: adapted from Bell and Grimson, 1992.



Typical Heterogeneous Environment

Major Objectives

- Location Transparency
  - User does not have to know the location of the data
  - Data requests automatically forwarded to appropriate sites
- Local Autonomy
  - Local site can operate with its database when network connections fail
  - Each site controls its own data, security, logging, recovery

Significant Trade-Offs

- **Synchronous** Distributed Database
  - All copies of the same data are always identical
  - Data updates are immediately applied to all copies throughout network
  - Good for data integrity
  - High overhead ➔ slow response times
- **Asynchronous** Distributed Database
  - Some data inconsistency is tolerated
  - Data update propagation is delayed
  - Lower data integrity
  - Less overhead ➔ faster response time

Advantages of Distributed Database over Centralized Databases

- Increased reliability/availability
- Local control over data
- Modular growth
- Lower communication costs
- Faster response for certain queries

Disadvantages of Distributed Database Compared to Centralized Databases

- Software cost and complexity
- Processing overhead
- Data integrity exposure
- Slower response for certain queries

Options for Distributing a Database

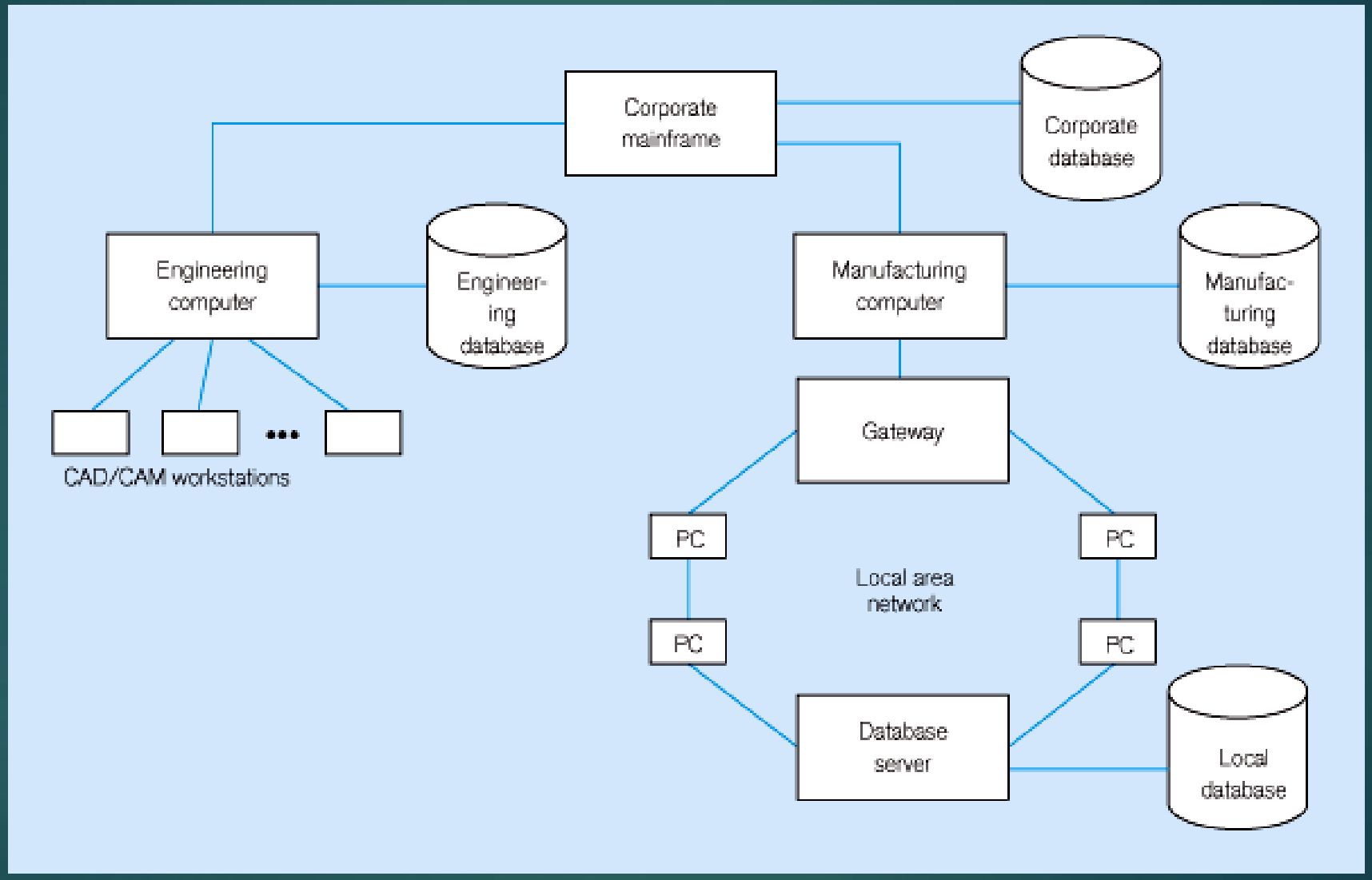
- Data replication
  - Copies of data distributed to different sites
- Horizontal partitioning
  - Different rows of a table distributed to different sites
- Vertical partitioning
  - Different columns of a table distributed to different sites
- Combinations of the above

- Advantages:
  - Reliability
  - Fast response
  - May avoid complicated distributed
  - transaction integrity routines (if replicated data is refreshed at scheduled intervals)
  - Decouples nodes (transactions proceed even if some nodes are down)
  - Reduced network traffic at prime time (if updates can be delayed)

Issues for Data Replication

- Data timeliness – high tolerance for out-of-date data may be required
- DBMS capabilities – if DBMS cannot support multi-node queries, replication may be necessary
- Performance implications – refreshing may cause performance problems for busy nodes
- Network heterogeneity – complicates replication
- Network communication capabilities – complete refreshes place heavy demand on telecommunications

Distributed processing system for a manufacturing company



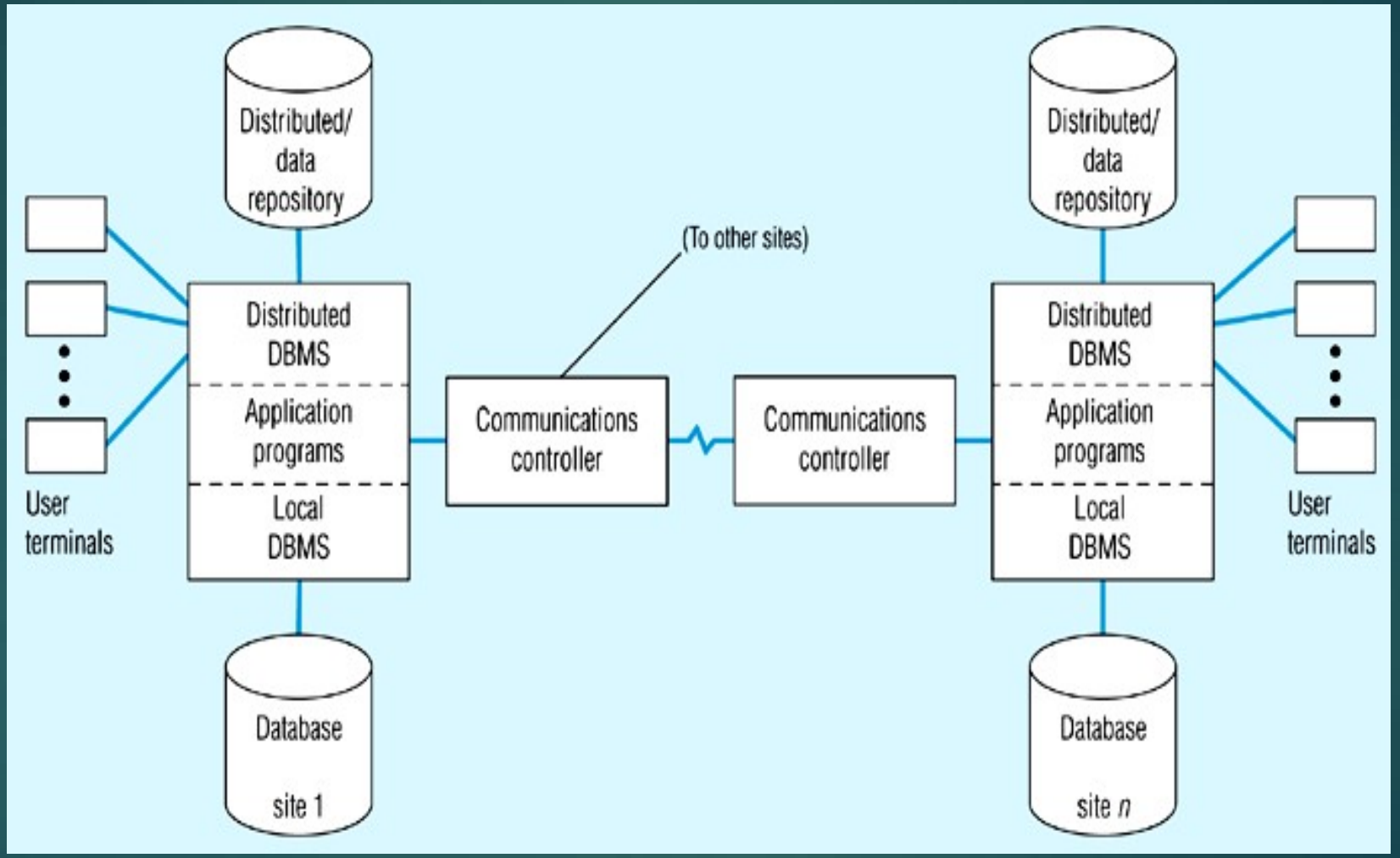
Five Distributed Database Organizations

- Centralized database, distributed access
- Replication with periodic snapshot update
- Replication with near real-time synchronization of updates
- Partitioned, one logical database
- Partitioned, independent, nonintegrated segments

Distributed DBMS

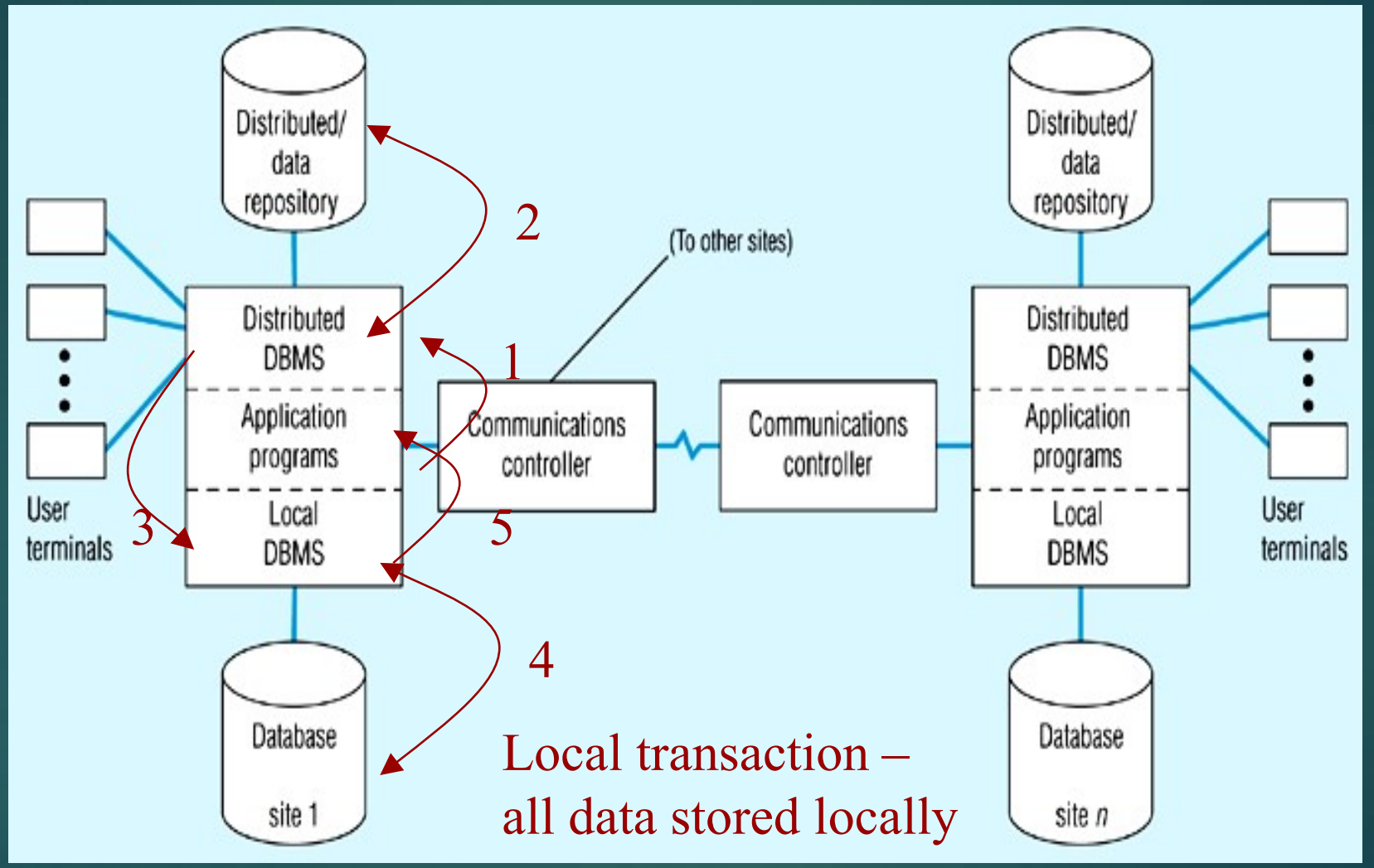
- ***Distributed database*** requires ***distributed DBMS***
- Functions of a distributed DBMS:
  - Locate data with a *distributed data dictionary*
  - Determine location from which to retrieve data and process query components
  - DBMS translation between nodes with different local DBMSs (using *middleware*)
  - Data consistency (via *multiphase commit protocols*)
  - Global primary key control
  - Scalability
  - Security, concurrency, query optimization, failure recovery

Figure 13-10: Distributed DBMS architecture



Local Transaction Steps

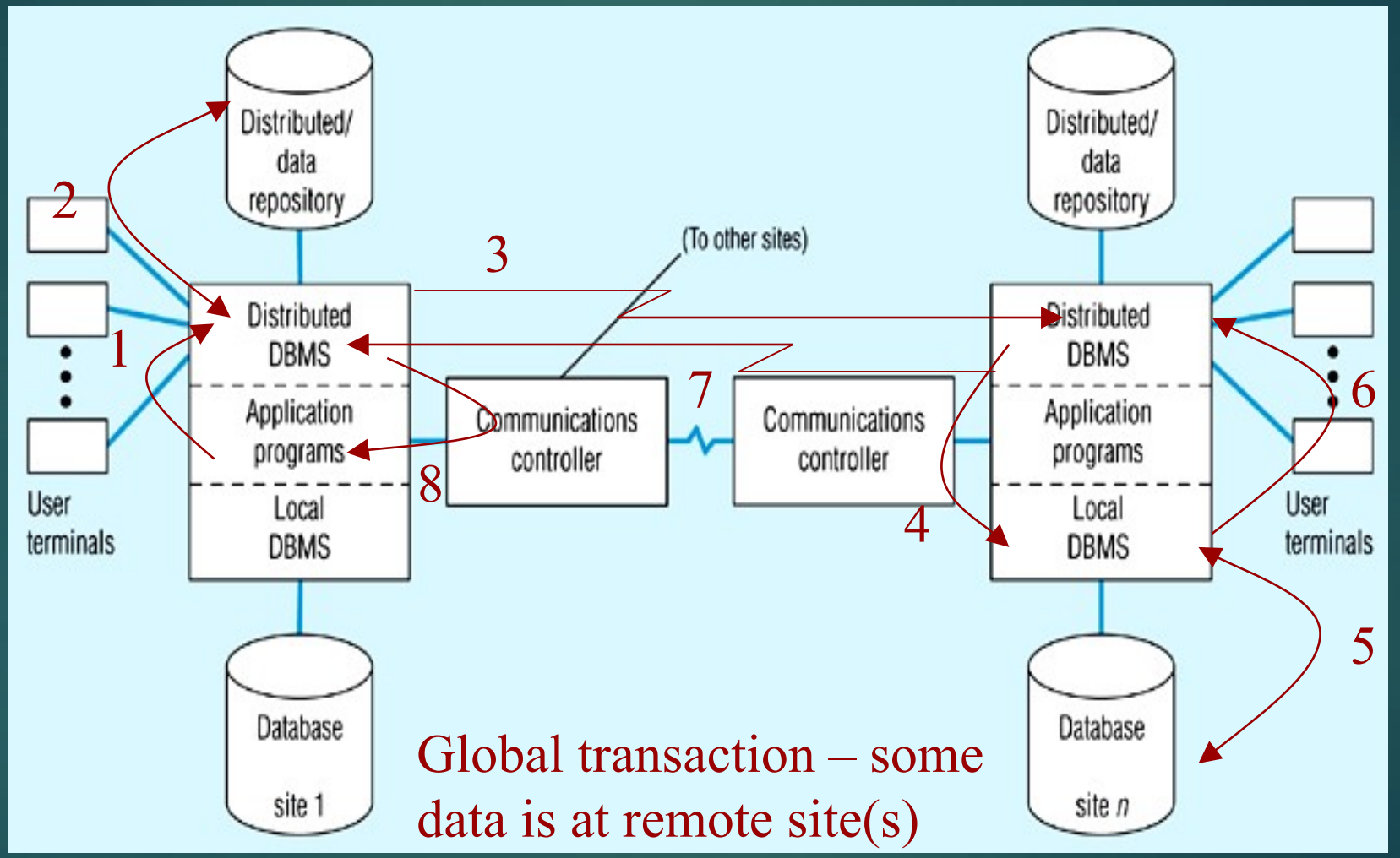
1. Application makes request to distributed DBMS
2. Distributed DBMS checks distributed data repository for location of data. Finds that it is **local**
3. Distributed DBMS sends request to local DBMS
4. Local DBMS processes request
5. Local DBMS sends results to application



Distributed DBMS Architecture (cont.) (showing local transaction steps)

Global Transaction Steps

1. Application makes request to distributed DBMS
2. Distributed DBMS checks distributed data repository for location of data. Finds that it is **remote**
3. Distributed DBMS routes request to remote site
4. Distributed DBMS at remote site translates request for its local DBMS if necessary, and sends request to local DBMS
5. Local DBMS at remote site processes request
6. Local DBMS at remote site sends results to distributed DBMS at remote site
7. Remote distributed DBMS sends results back to originating site
8. Distributed DBMS at originating site sends results to application



showing global transaction steps

Distributed DBMS Transparency Objectives

- Location Transparency
  - User/application does not need to know where data resides
- Replication Transparency
  - User/application does not need to know about duplication
- Failure Transparency
  - Either all or none of the actions of a transaction are committed
  - Each site has a transaction manager
    - Logs transactions and before and after images
    - Concurrency control scheme to ensure data integrity
  - Requires special *commit protocol*

Two-Phase Commit

- ***Prepare Phase***
  - Coordinator receives a commit request
  - Coordinator instructs all resource managers to get ready to "go either way" on the transaction. Each resource manager writes all updates from that transaction to its own physical log
  - Coordinator receives replies from all resource managers. If all are ok, it writes commit to its own log; if not then it writes rollback to its log
- ***Commit Phase***
  - Coordinator then informs each resource manager of its decision and broadcasts a message to either commit or rollback (abort). If the message is commit, then each resource manager transfers the update from its log to its database
  - A failure during the commit phase puts a transaction "in limbo." This has to be tested for and handled with timeouts or polling

Concurrency Control

- Concurrency Transparency
  - Design goal for distributed database
- Timestamping
  - Concurrency control mechanism
  - Alternative to locks in distributed databases

- Thank You