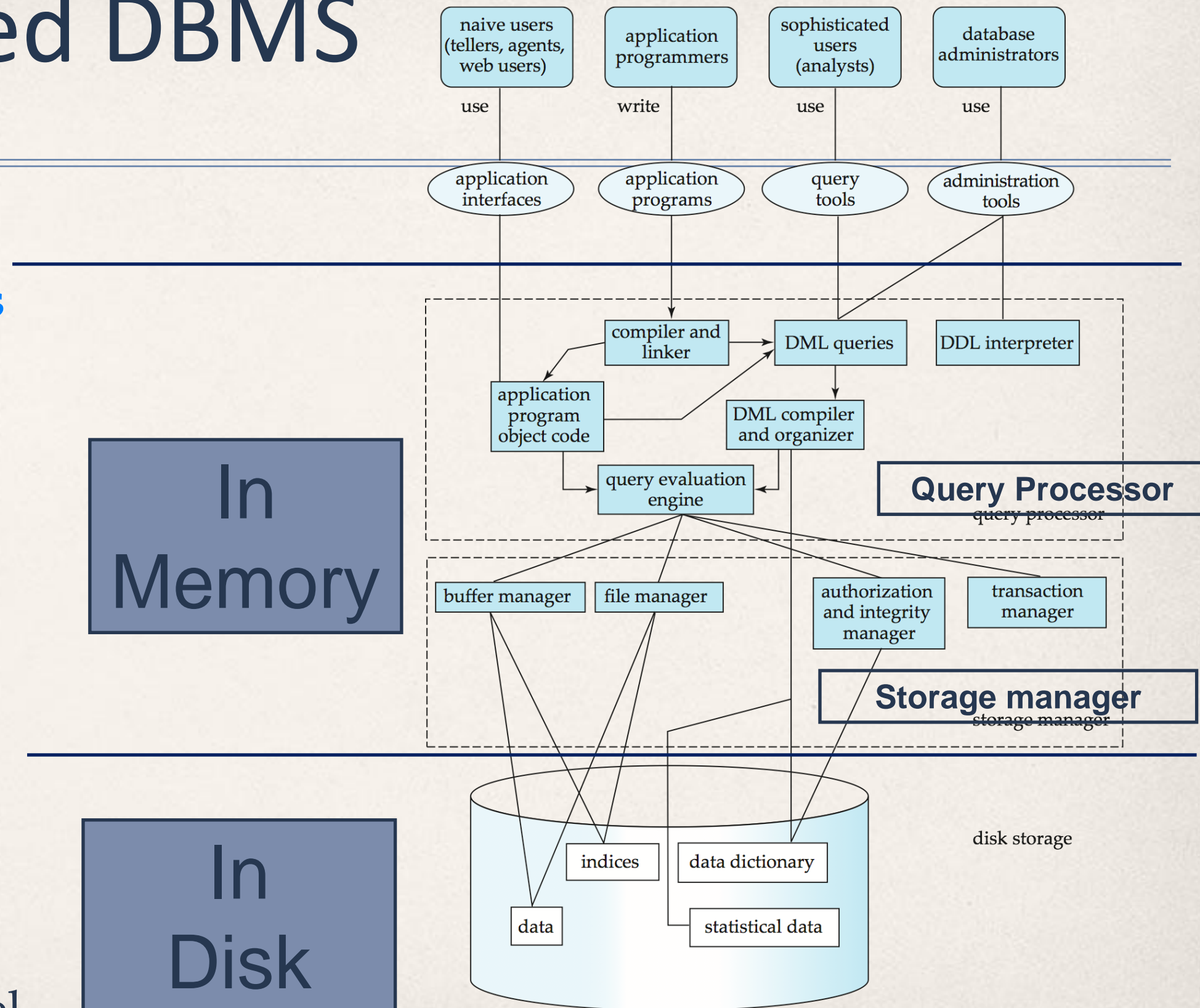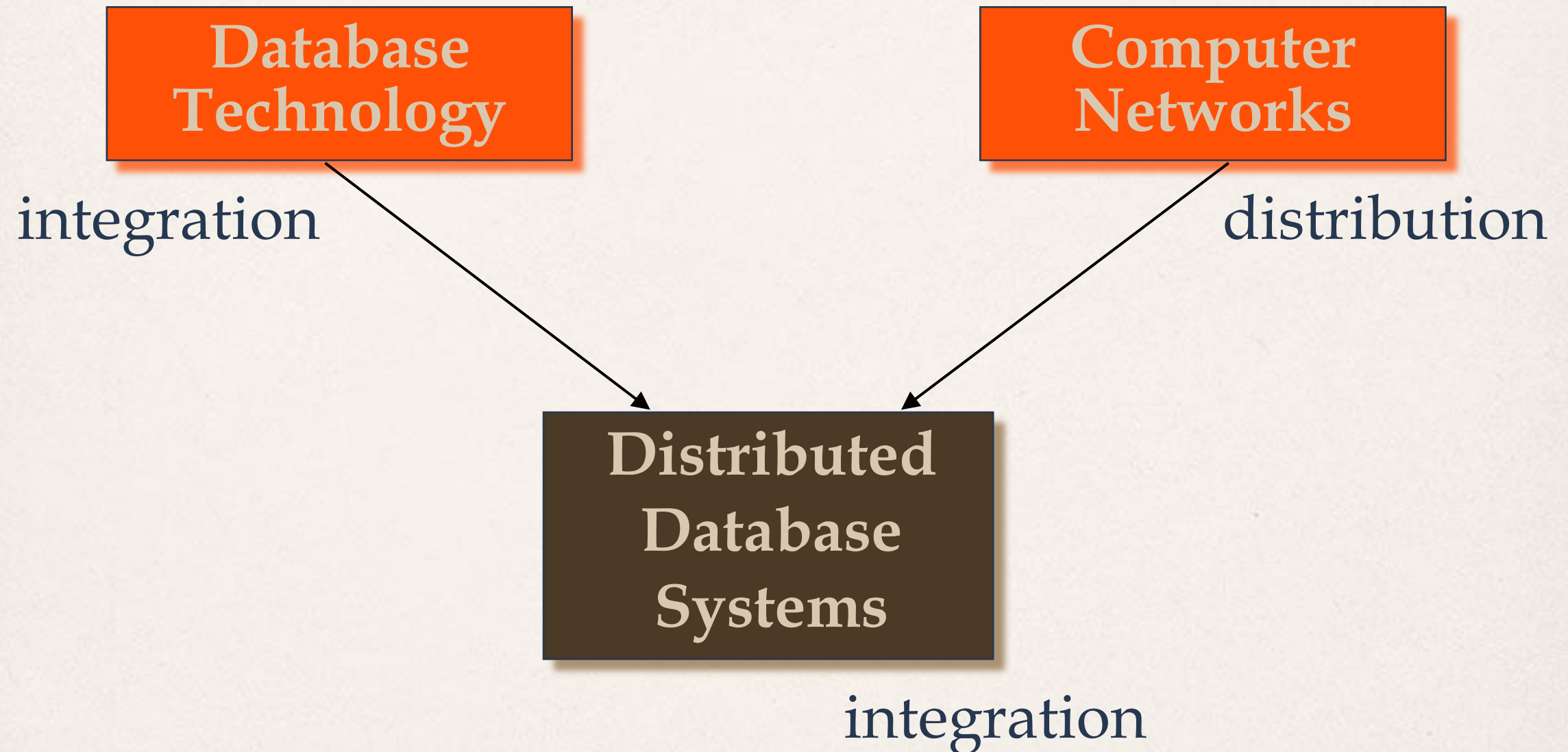# Distributed Database

Centralized DBMS

Main Components
1. Data in storage
2. Querying Data
3. Query Optimization
4. Indexing
5. Transaction processing
6. Application Development and Design
7. Architecture
8. Advanced Model

naive users (tellers, agents, web users) — use
application programmers — write
sophisticated users (analysts) — use
database administrators — use

application interfaces
application programs
query tools
administration tools

compiler and linker
DML queries
DDL interpreter

application program object code
DML compiler and organizer

query evaluation engine

Query Processor
query processor

In Memory

buffer manager
file manager
authorization and integrity manager
transaction manager

Storage manager
storage manager

In Disk

indices
data dictionary
data
statistical data

disk storage

Motivation

Database Technology

Computer Networks

integration

distribution

Distributed Database Systems

integration

integration ≠ centralization

What is a Distributed Database System?

A distributed database (DDB) is a collection of multiple, *logically interrelated* databases distributed over a *computer network.*
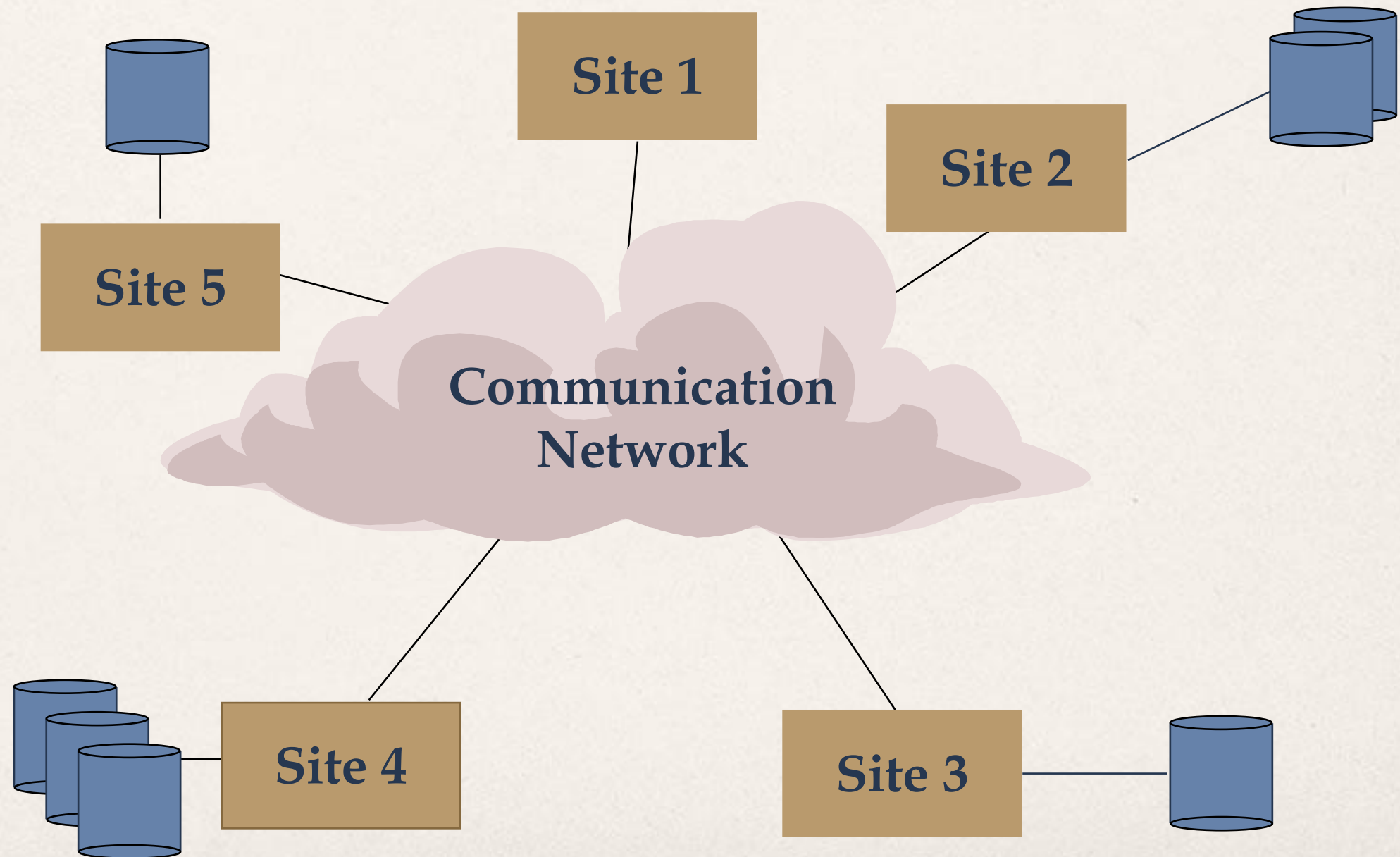
A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.

Distributed database system (DDBS) = DDB + D–DBMS
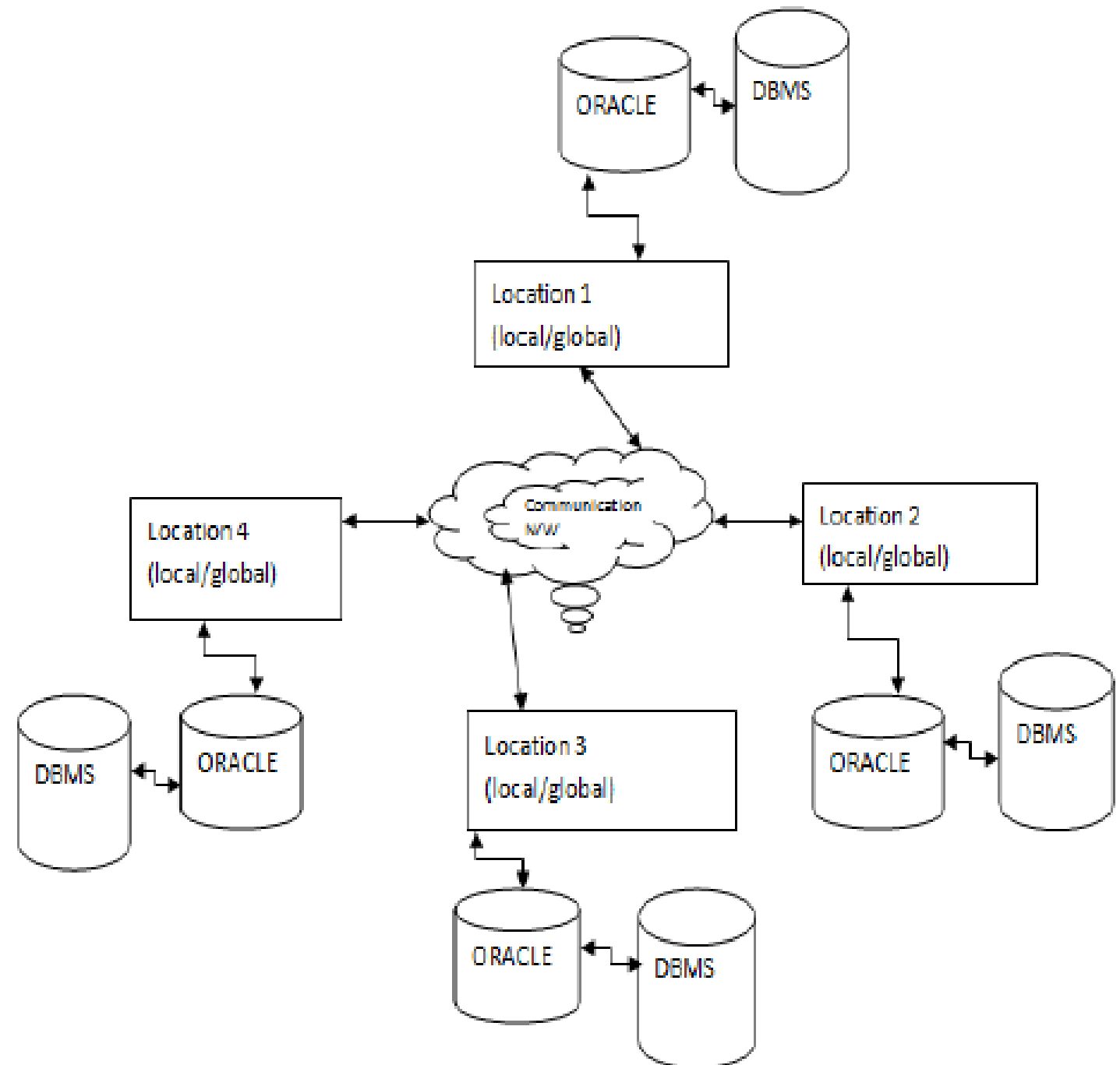
Centralized DBMS on a Network
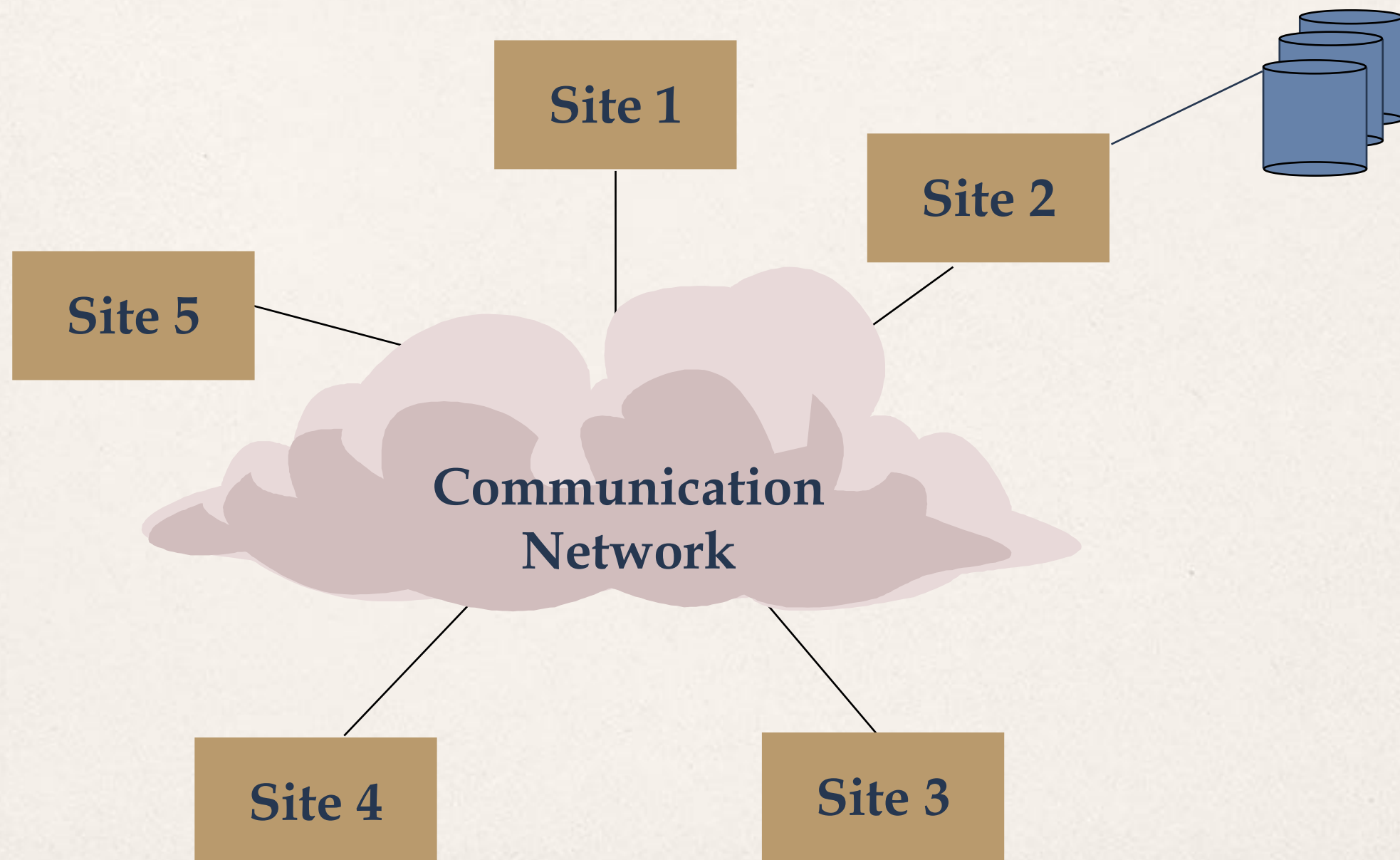
Distributed DBMS Environment

Homogeneous Distributed Database

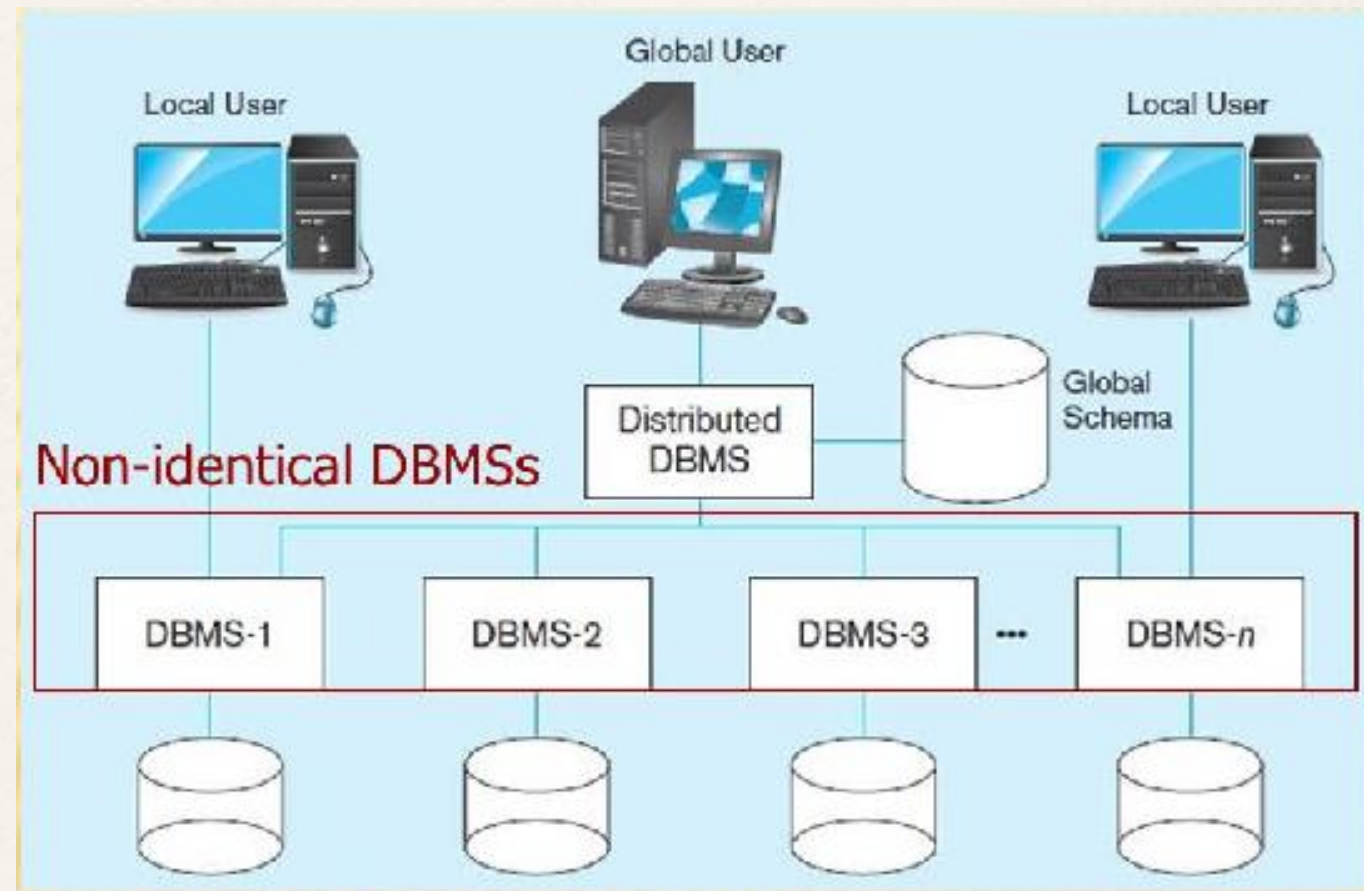In Homogeneous Distributed Database –

All sites shall have same DBMS

Heterogeneous Distributed Database

In Heterogeneous Distributed Database –

Different sites shall have different DBMS

Question 19-1: Compare homogeneous and heterogeneous DDBMS in terms of storage, querying and transaction.

Transparency

- Transparency is the separation of the higher level semantics of a system from the lower level implementation issues.

- Fundamental issue is to provide

  data independence

  in the distributed environment

  - → Network (distribution) transparency

  - → Replication transparency

  - → Fragmentation transparency
    - ✦ horizontal fragmentation: selection
    - ✦ vertical fragmentation: projection
    - ✦ hybrid

Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

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

Transparent Access

```
SELECT  ENAME,SAL
FROM    EMP,ASG,PAY
WHERE   DUR > 12
AND     EMP.ENO = ASG.ENO
AND     PAY.TITLE = EMP.TITLE
```


Tokyo

Paris

Paris projects
Paris employees
Paris assignments
Boston employees

Boston

Communication
Network

Boston projects
Boston employees
Boston assignments

Montreal

Montreal projects
Paris projects
New York projects
with budget > 200000
Montreal employees
Montreal assignments

New
York

Boston projects
New York employees
New York projects
New York assignments

Reliability Through Distributed Transactions

- Replicated components and data should make distributed DBMS more reliable.
- Distributed transactions provide
  - → Concurrency transparency
  - → Failure atomicity
- Distributed transaction support requires implementation of
  - → Distributed concurrency control protocols
  - → Commit protocols
- Data replication
  - → Great for read-intensive workloads, problematic for updates
  - → Replication protocols

Potentially Improved Performance

- Proximity of data to its points of use

  → Requires some support for fragmentation and replication

- Parallelism in execution

  → Inter-query parallelism

  → Intra-query parallelism

Parallelism Requirements

- Have as much of the data required by *each* application at the site where the application executes

  → Full replication   Read-intensive: more replication, better
Write-intensive: less replication, better

- How about updates?

  → Mutual consistency

  → Freshness of copies

Distributed DBMS Issues

- **Distributed Database Design**
  - → How to distribute the database
  - → Replicated & non-replicated database distribution
  - → A related problem in directory management
- **Query Processing**
  - → Convert user transactions to data manipulation instructions
  - → Optimization problem
    - ✦ min{cost = data transmission + local processing}
  - → General formulation is NP-hard

# Distributed DBMS Issues

- **Concurrency Control**

  → Synchronization of concurrent accesses

  → Consistency and isolation of transactions' effects

  → Deadlock management

- **Reliability**

  → How to make the system resilient to failures

  → Atomicity and durability

Question 19-2: Explain the challenges in data transparency and transactional reliability in DDBMS.

Relationship Between Issues
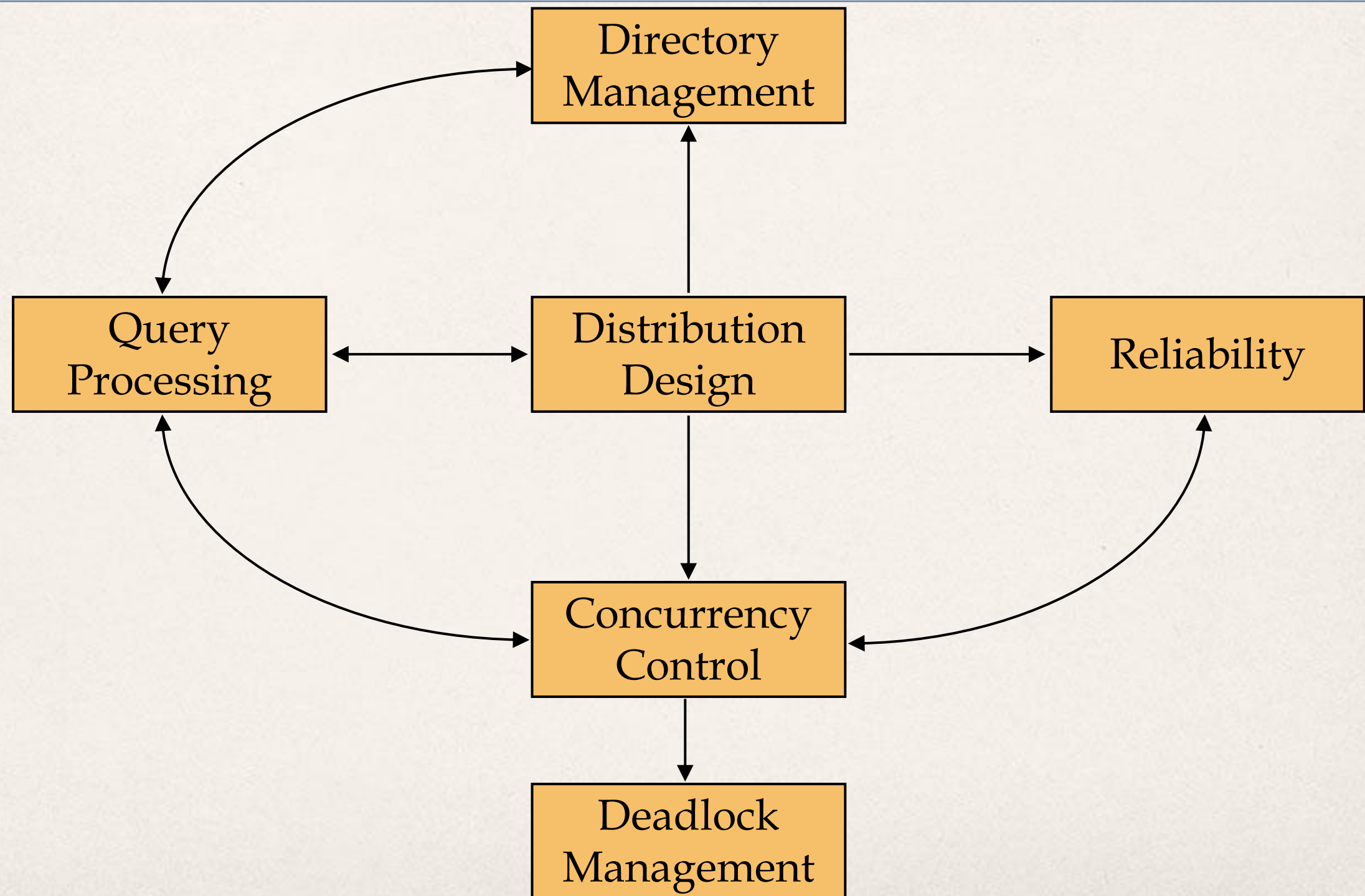
Mediator/Wrapper Architecture for Data Integration