CHAPTER 4

DISTIBUTED DATABASE SYTEM /DDBS/

CHAPTER OUTLINE

1. Concepts of Distributed databases
2. Distributed database design
3. Distributed query processing
4. Distributed transaction management and recovery

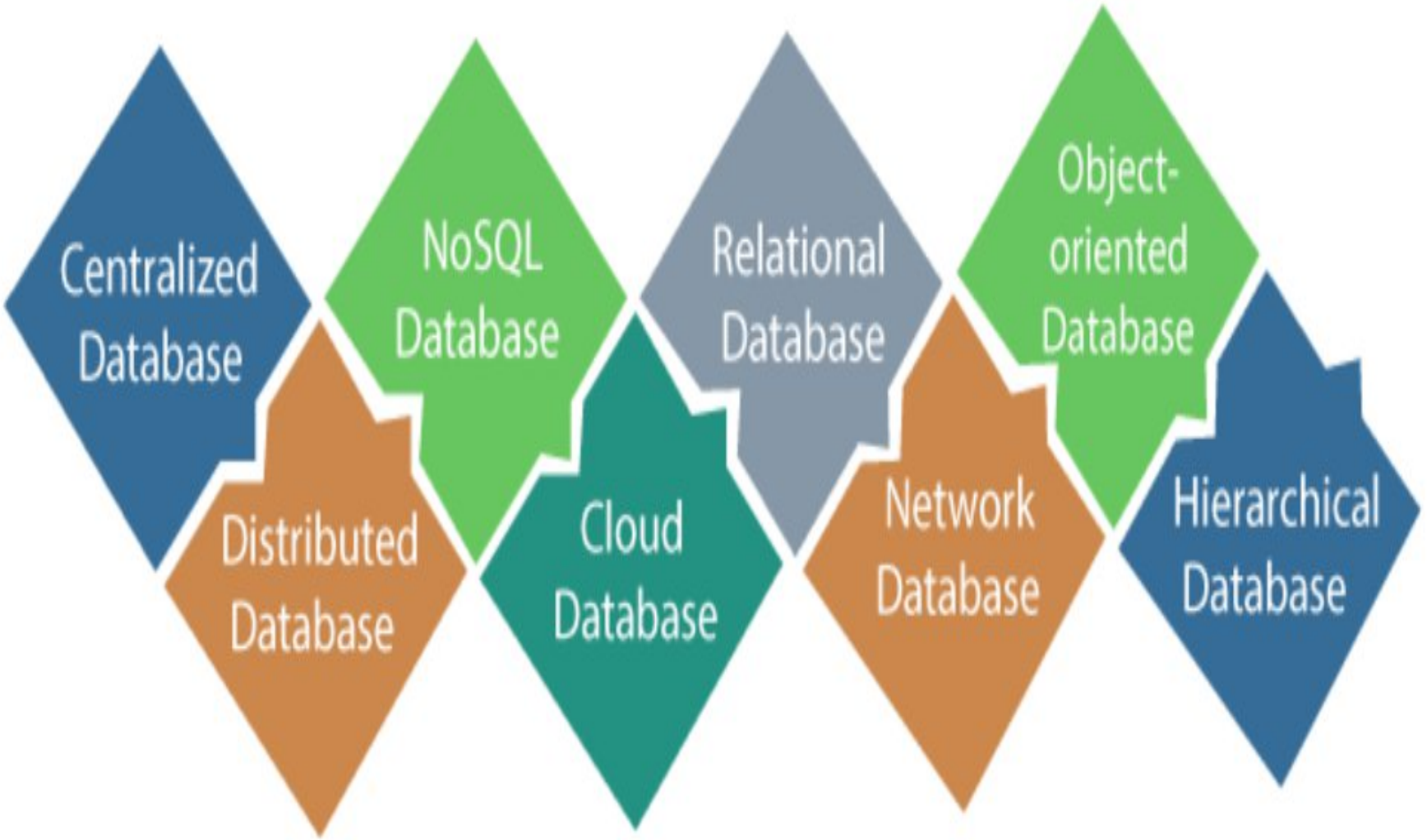
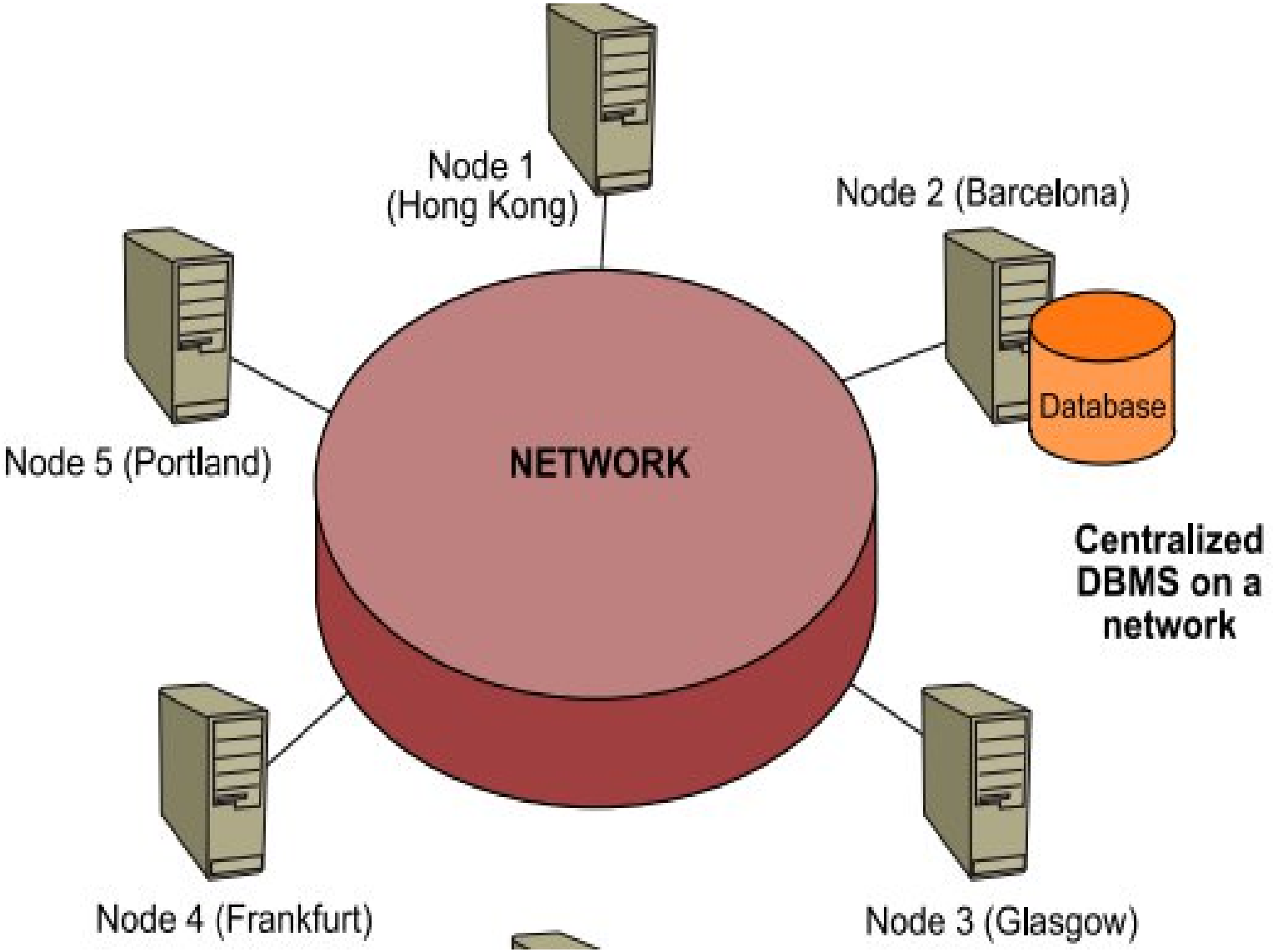
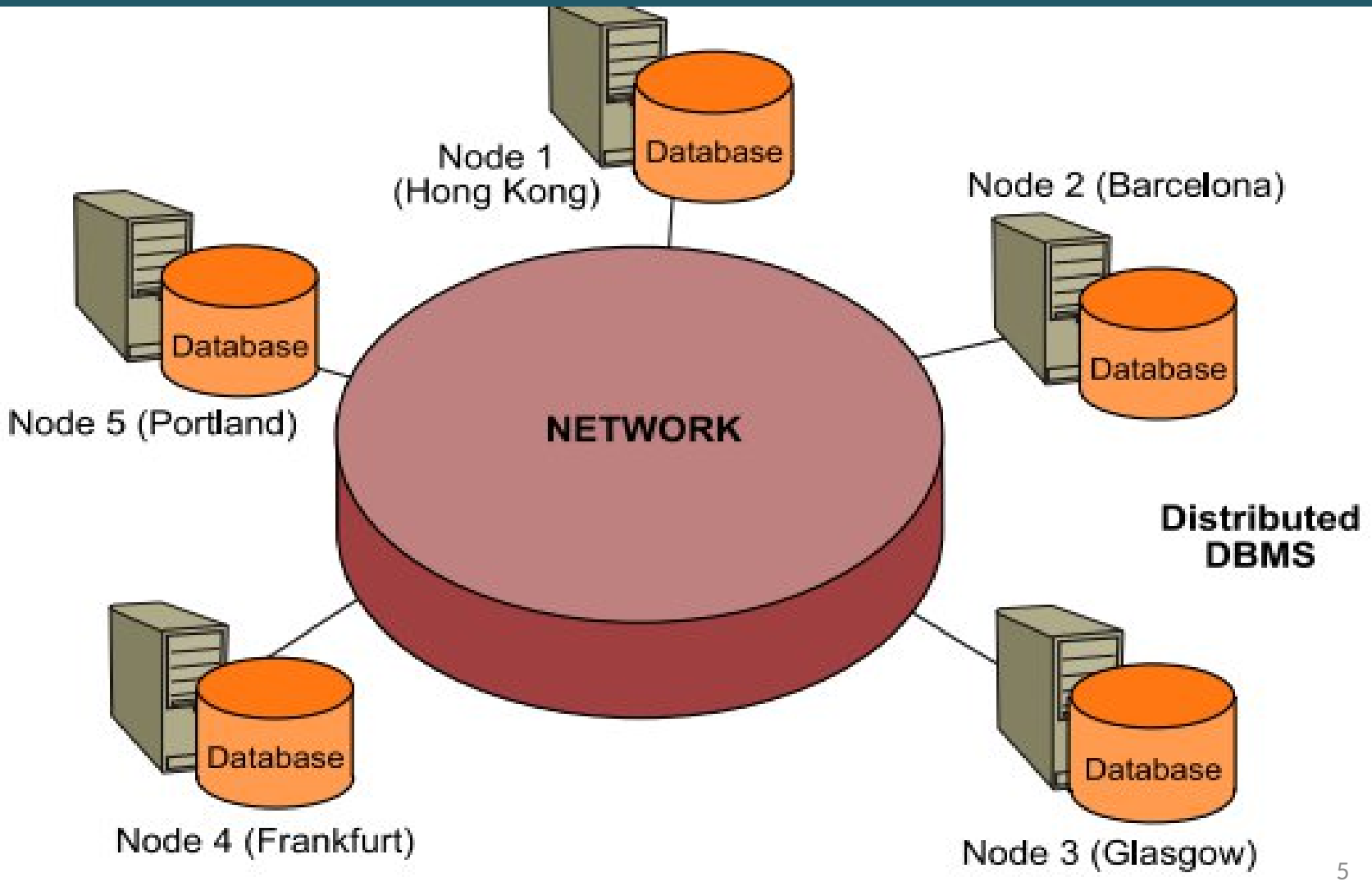


Figure 1: Types of Databases

Central DB


Node 1
(Hong Kong)
Node 2 (Barcelona)
Database
Node 5 (Portland)
NETWORK
Centralized
DBMS on a
network
Node 4 (Frankfurt)
Node 3 (Glasgow)

DISTIBUTED DATABASE SYSTEM

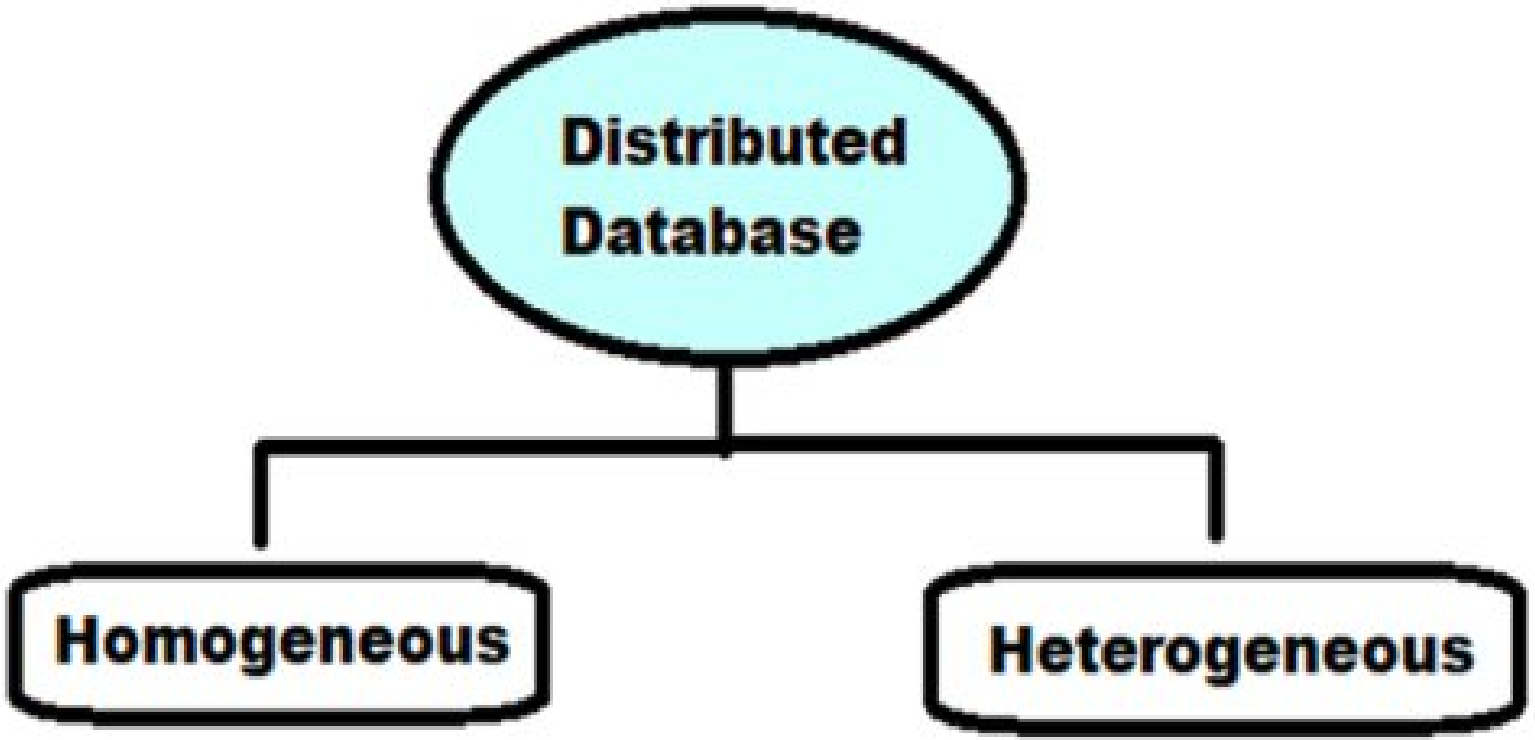


Distributed Database system

- Distributed database is defined as is a collection of multiple, logically interrelated databases that are spread across different locations but connected through a network.
- Even though the data is stored in different physical sites, the system works as **one single database** to the user.
- A distributed database management system (DDBMS) is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users.
- The two important terms in these definitions are “logically interrelated” and “distributed over a computer network.”
- Unlike a centralized database that runs on a single machine, a distributed database appears as a single logical database to the user, but its data is physically distributed across multiple sites connected by a network.

- **Example:** A bank with branches in many cities where each branch stores its own customers' data, but all branches can access each other's data as one database.

Types of DDB



Homogenous distributed database system

- in a homogenous distributed database system, all the physical locations have the same underlying hardware and run the same operating systems and database applications. i.e on all computers Oracle/sql/other DBMS is used as DBMS system.
- It appear to the user as a single system, and they can be much easier to design and manage.
- For a distributed database system to be homogenous, the data structures at each location must be either identical or compatible.
- The database application used at each location must also be either identical or compatible.

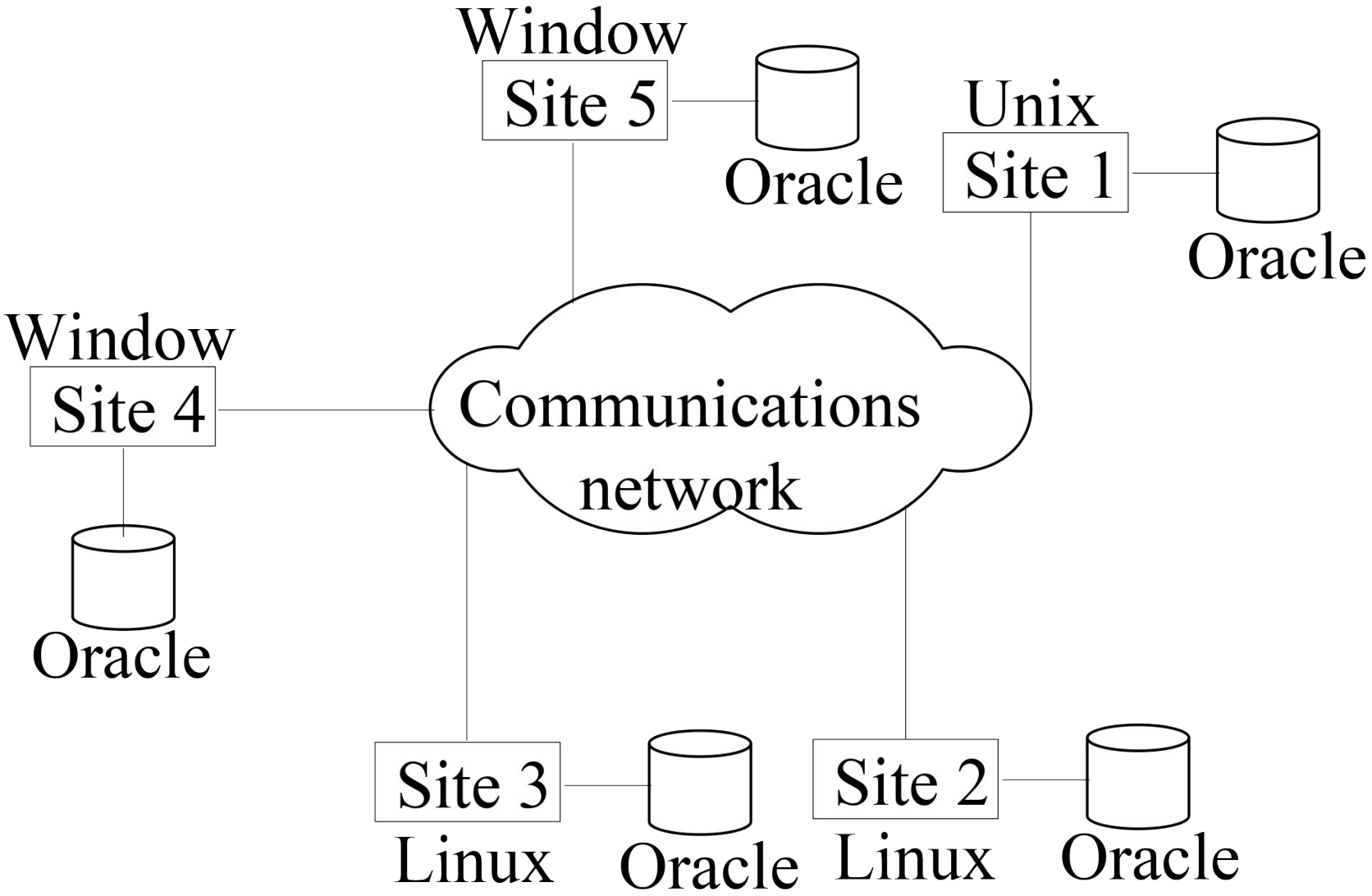
There are two types of homogeneous distributed database:

- ✓ **Autonomous**: Each database is independent that is it functions on its own.

They are integrated by a controlling application and use message passing to share data updates.

- ✓ **Non-autonomous**: Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites

Example of Homo DDB



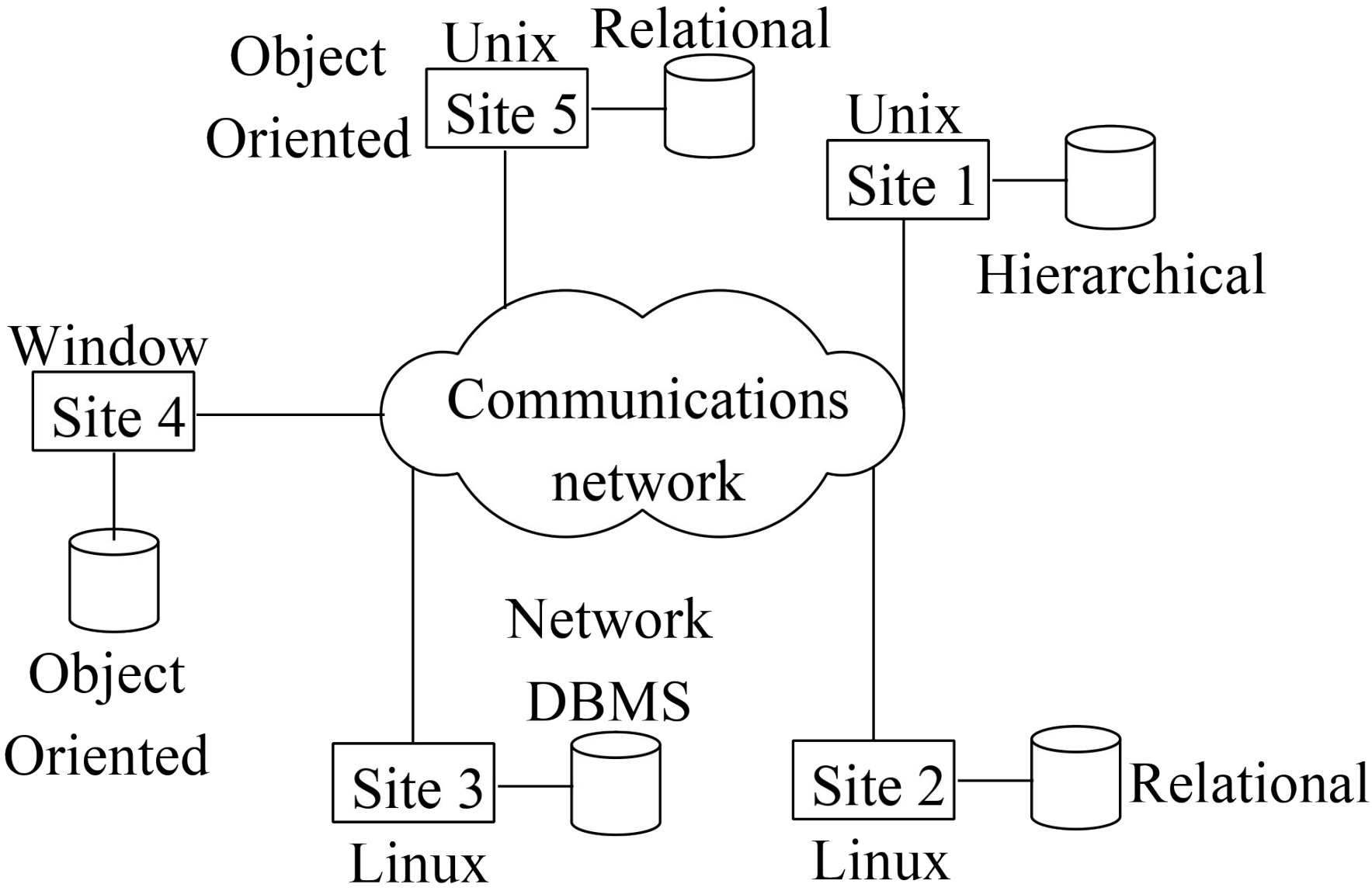
Heterogeneous Distributed Database System

- In a heterogeneous distributed database, the hardware, operating systems or database applications may be different at each location.
- Different sites may use different schemas and software, although a difference in schema can make query and transaction processing difficult.
- Different nodes may have different hardware, software and data structure, or they may be in locations that are not compatible.
- Users at one location may be able to read data at another location but not upload or alter it.
- Heterogeneous distributed databases are often difficult to use, making them economically infeasible for many businesses.

---Types Hetro DDB

- **Federated**: The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated**: The database systems employ a central coordinating module through which the databases are accessed.

Example of HDDBS



❑ Advantages of DDB :

Improved Reliability & Availability

Failure of one site doesn't crash the entire system.

Data replication allows continued access from other nodes.

Scalability

Easier to scale horizontally by adding more servers/nodes.

Can handle increased load by distributing data and transactions.

Performance & Speed

Data can be stored closer to where it's most frequently used (location transparency).

Parallel processing of queries across nodes reduces response times.

- Easier expansion (scalability):
- Allows new nodes (computers) to be added anytime without changing the entire configuration.
- The main advantage of a distributed database system is that it can provide higher availability and reliability than a centralized database system.
- Because the data is stored across multiple sites, the system can continue to function even if one or more sites fail.
- In addition, a distributed database system can provide better performance by distributing the data and processing load across multiple sites.

❑ Disadvantages of Distributed Database

Complexity- Design, implementation, and maintenance are significantly more complex than centralized DBS.

Requires handling network issues, partitioning, replication, and distributed transactions.

Cost- Since DDBMS needs more people and more hardware , maintaining and running the system can be more expensive than the centralized system .

Problem of connecting Dissimilar Machine- Additional layers of operation system software are needed to translate and coordinate the flow of data between machines.

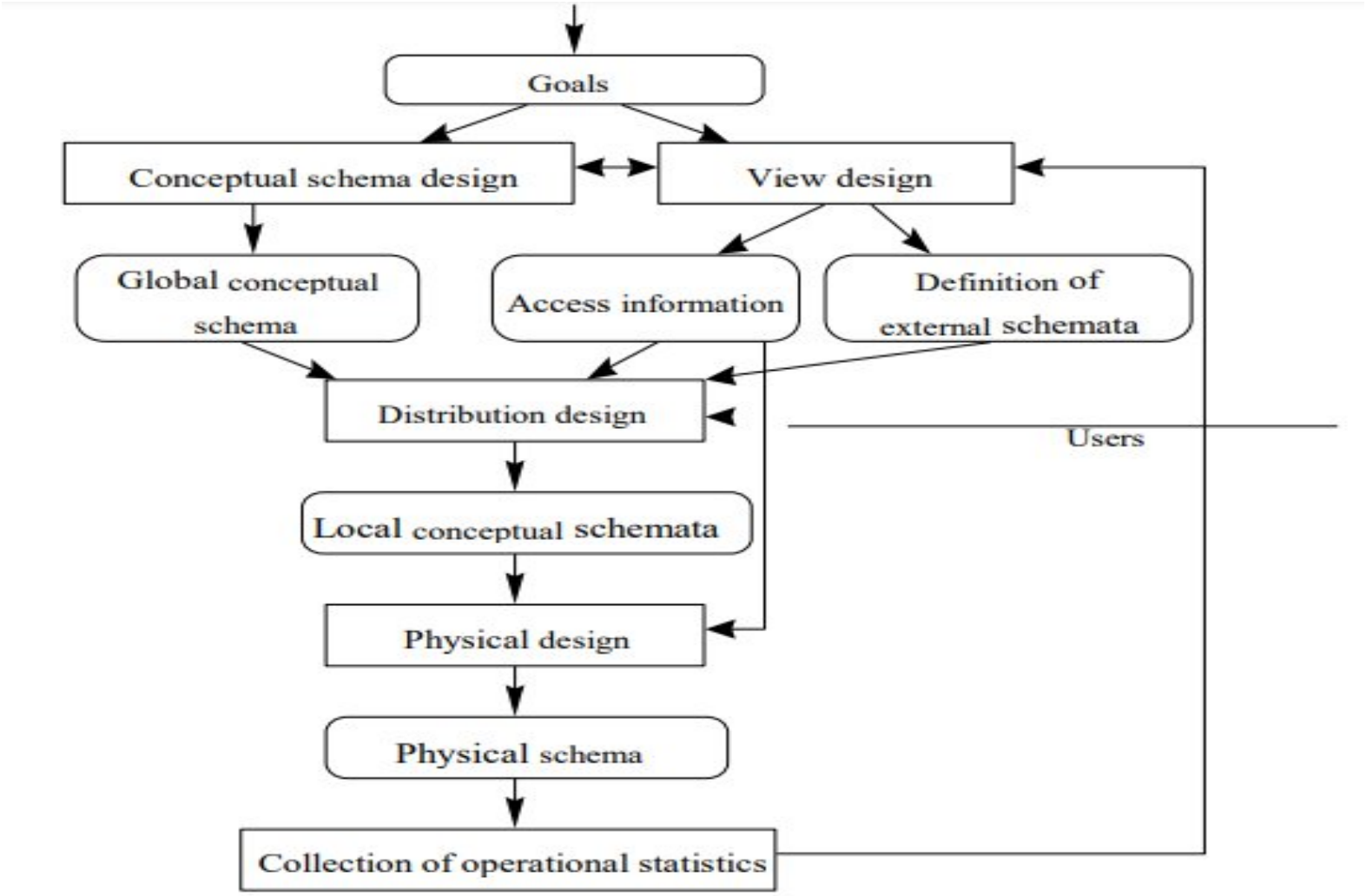
Data integrity and security problem - Because data maintained by distributed systems can be accessed at locations in the network, controlling the integrity of a database can be difficult.

Distributed database design

There are in general several design alternatives. But common are:

- Top-down approach: first the general concepts, the global framework are defined, after then the details.
- Down-top approach: first the detail modules are defined, after then the global framework.
- If the system should match to existing systems or some modules are yet ready, the down-top method is usually used.

Top- Down design example



General design steps according to the structure

- ✓ analysis of the external, application requirements
- ✓ design of the global schema
- ✓ design of the fragmentation
- ✓ design of the distribution schema
- ✓ design of the local schemes
- ✓ design of the local physical layers

DDBMS -specific design steps: -

- ✓ design of the fragmentation
- ✓ design of the distribution schema
- ✓ - During the requirement analysis phase, also the fragmentation and distribution requirements are considered.

2. Data Replication and Fragmentation: *Distributed data storage*

- There are two approaches to store the relation in the distributed database : **Replication and Fragmentation**

I. Data Replication:-

- The system maintain several identical copies of the relation & store each copy at a different site
- In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead

II. Data Fragmentation

- Split a relation into logically related and correct parts.
- **Data Fragmentation** is the process of splitting a database into smaller subsets called **fragments** that are stored on different nodes.
- This is primarily done for **scalability and performance.**

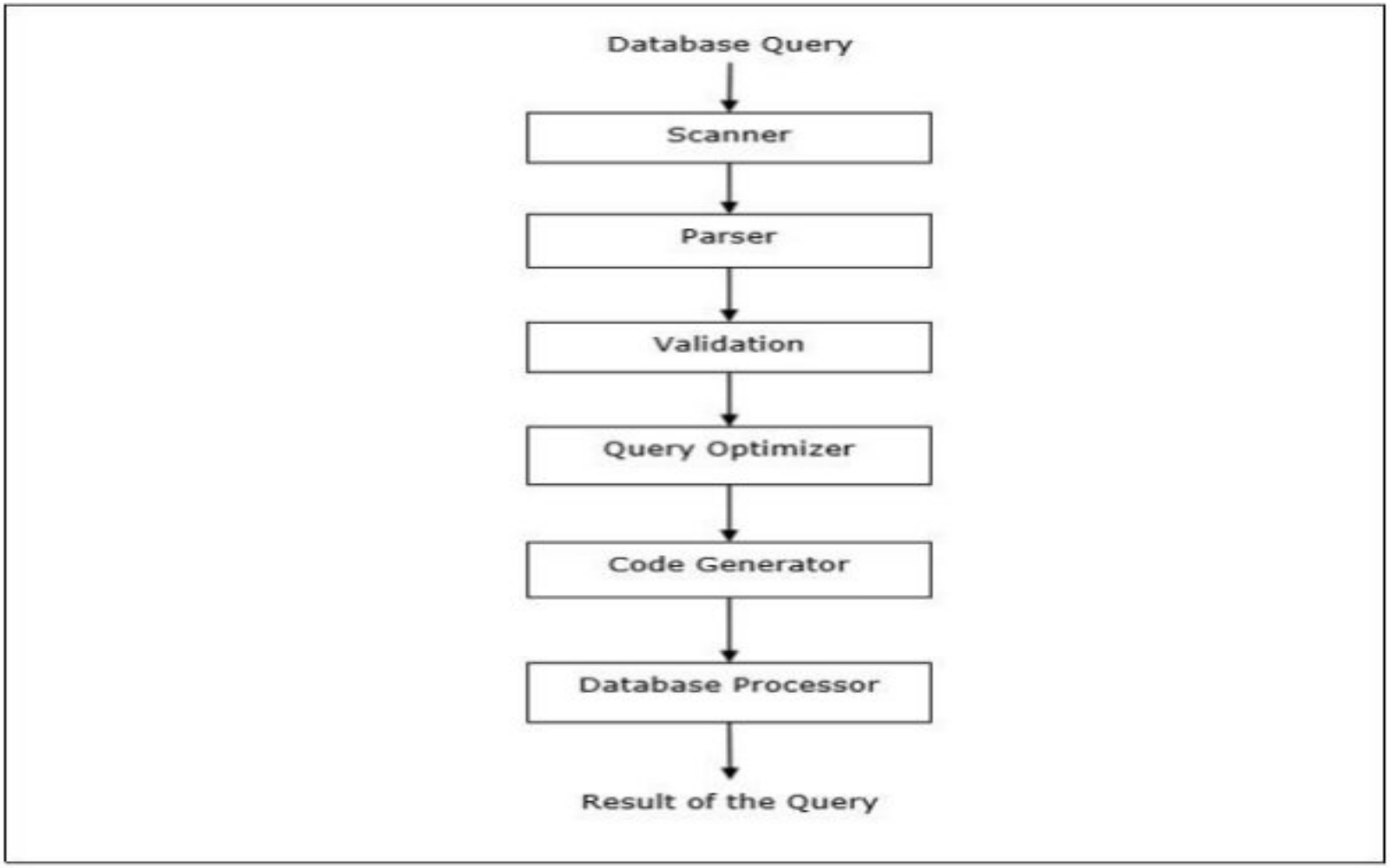
- A relation can be fragmented in two ways:
  - **Horizontal fragmentation**
    - ✓ It is a horizontal subset of a relation which contain those of rows which satisfy selection conditions.
    - ✓ Splitting a table by ROWS
    - ✓ Consider the Employee relation with selection condition (DNO = 5). All rows satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.
    - ✓ A selection condition may be composed of several conditions connected by AND or OR.

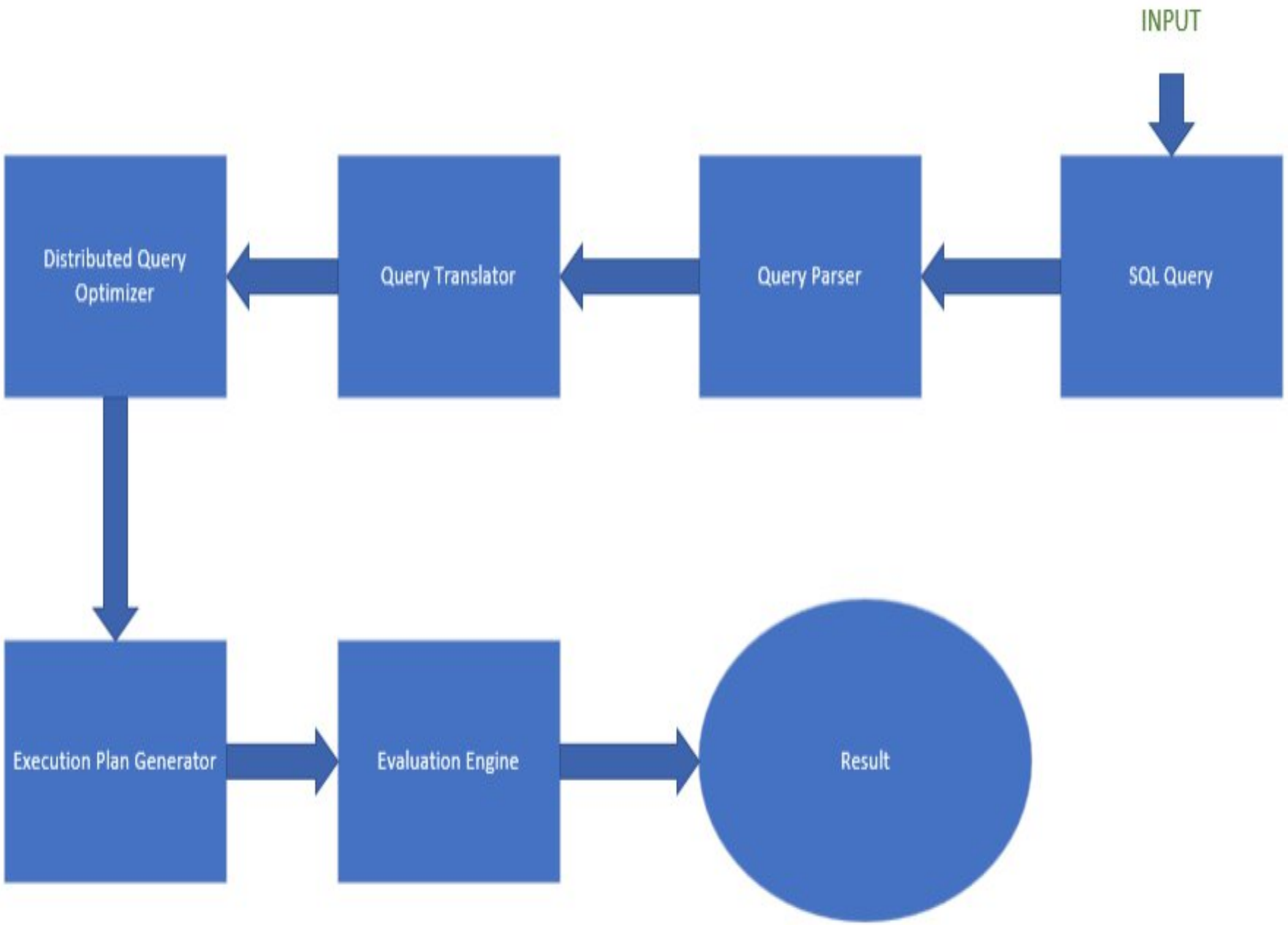
Vertical fragmentation

- It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns.
- Splitting a table by COLUMNS

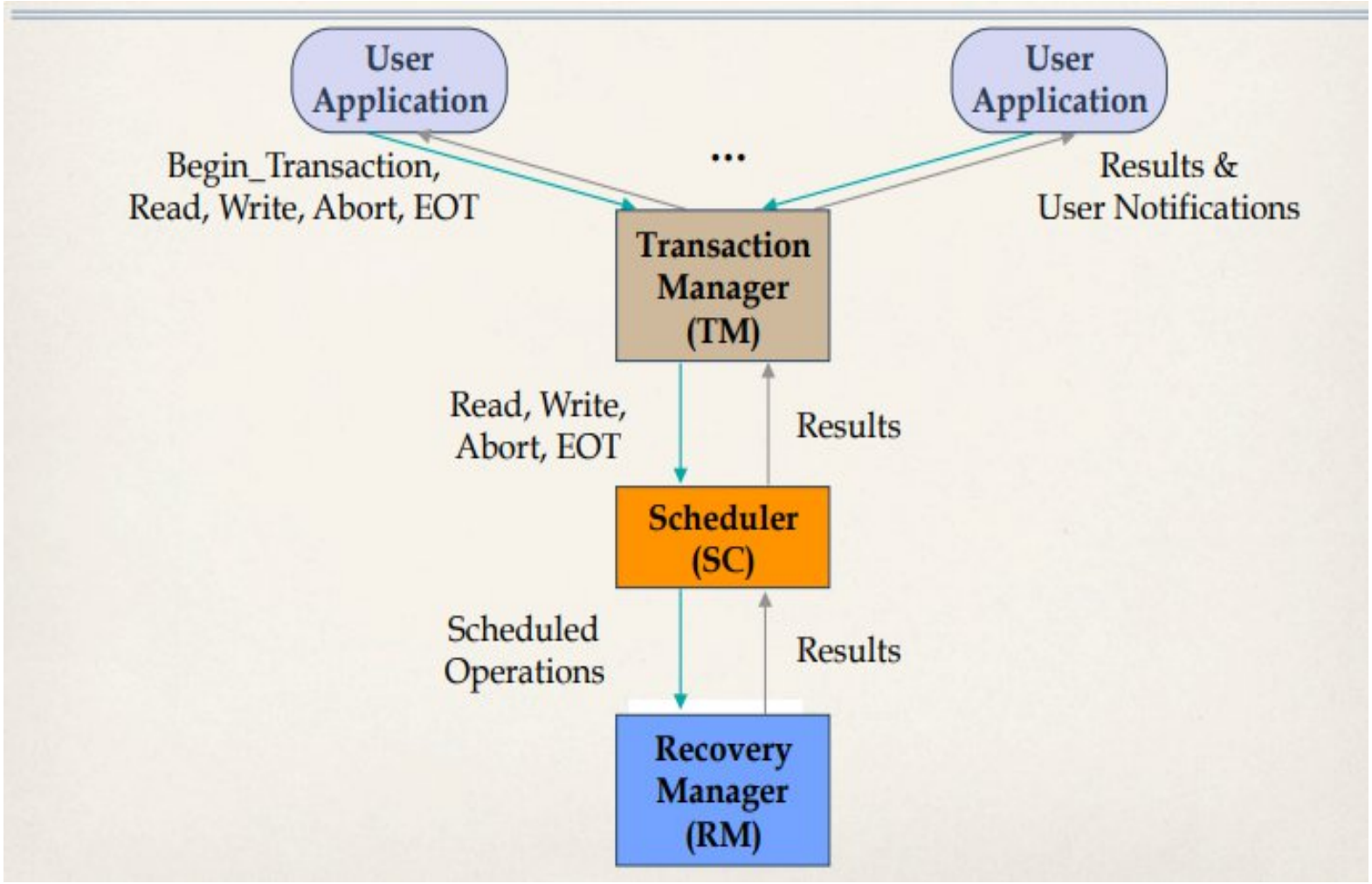
Distributed query processing

Query Processing Query processing is a set of all activities starting from query placement to displaying the results of the query. Steps are

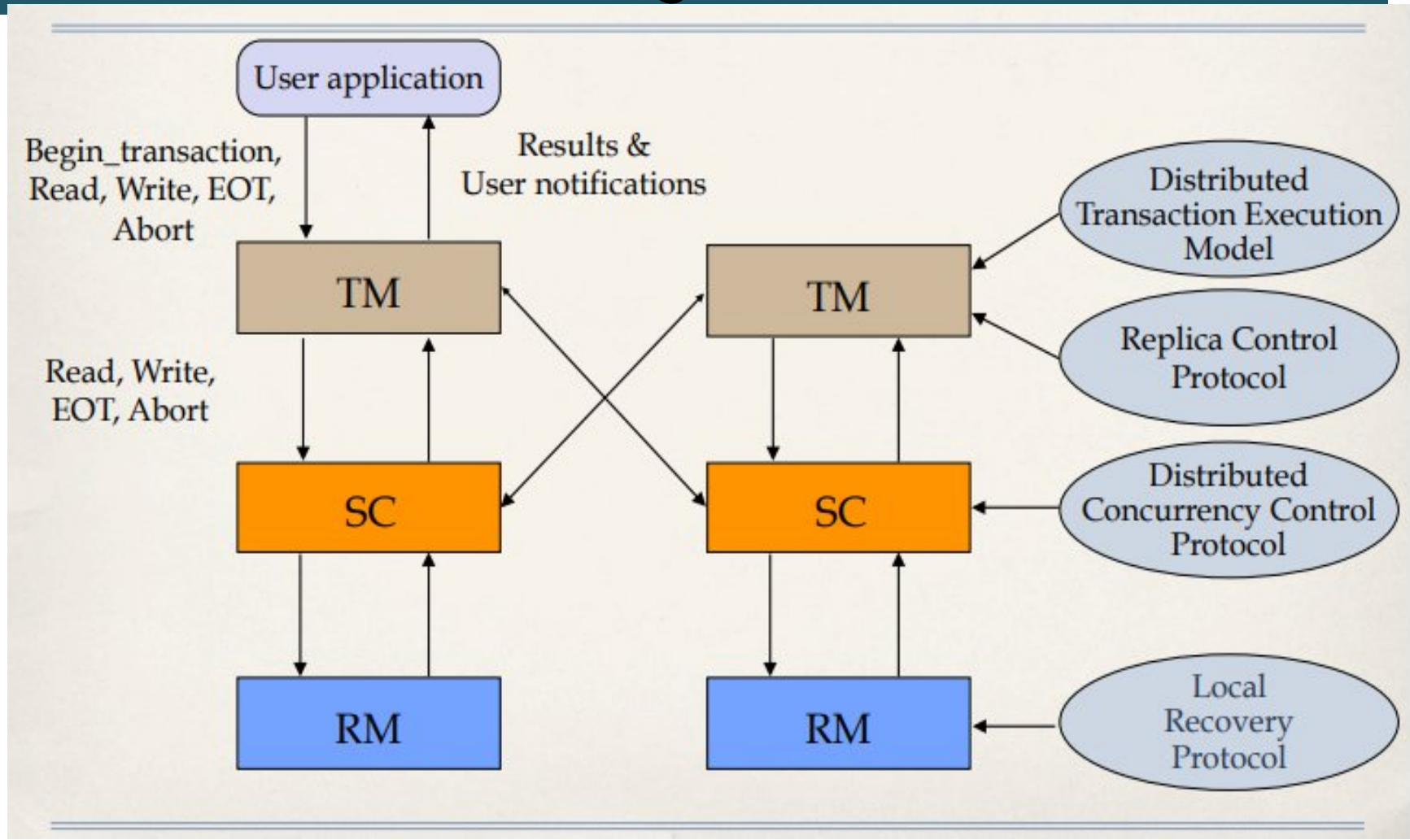



INPUT
SQL Query
Query Parser
Query Translator
Distributed Query Optimizer
Execution Plan Generator
Evaluation Engine
Result

Non Distributed /centralized transaction/



4. Distributed transaction execution and management



Distributed transaction recovery

- Recovery techniques in distributed systems are essential for ensuring that the system can return to a stable state after encountering errors or failures.
- Recovery in distributed systems focuses on maintaining functionality and data integrity despite failures.
- It involves strategies for detecting faults, restoring state, and ensuring continuity across interconnected nodes.

These techniques can be broadly categorized into the following:

- **Checkpointing**: Periodically saving the system's state to a stable storage, so that in the event of a failure, the system can be restored to the last known good state. Checkpointing is a key aspect of backward recovery.
- **Rollback Recovery**: Involves reverting the system to a previous checkpointed state upon detecting an error. This technique is useful for undoing the effects of

Cont..

- **Forward Recovery**: Instead of reverting to a previous state, forward recovery attempts to move the system from an erroneous state to a new, correct state. This requires anticipating possible errors and having strategies in place to correct them on the fly.
- **Logging and Replay**: Keeping logs of system operations and replaying them from a certain point to recover the system's state. This is useful in scenarios where a complete rollback might not be feasible.
- **Replication**: Maintaining multiple copies of data or system components across different nodes. If one component fails, another can take over, ensuring continuity of service.
- **Error Detection and Correction**: Incorporating mechanisms that detect errors and automatically correct them before they lead to system failure. This is a proactive approach that enhances system resilience.

Applications of DDB

- ✓ **Large-Scale Web Applications:** Websites and services like social media platforms, e-commerce sites, and content delivery networks use distributed databases to handle massive amounts of user-generated data and transactions. **Cloud Computing**: Many cloud services utilize distributed databases to provide scalable and reliable storage solutions.
- ✓ **Big Data Analytics**: Distributed databases are essential for big data applications, where data is collected from various sources and analyzed in real-time. Technologies like Apache Cassandra and Hadoop are often employed for such purposes
- ✓ **IoT Applications**: The Internet of Things (IoT) generates vast amounts of data from connected devices. Distributed databases can efficiently store and process this data while ensuring availability and fault tolerance.

Cont..

- ✓ **Financial Services**: Banks and financial institutions use distributed databases to maintain transaction records across multiple branches and locations, ensuring data consistency and security.
- ✓ **Healthcare**: Distributed databases can manage patient records and medical data across different healthcare facilities, enabling better data sharing and collaboration while maintaining compliance with regulations.
- ✓ **Telecommunications**: Telecom companies utilize distributed databases to handle call records, billing information, and customer data across various regions and networks.

?