Distributed Databases

Distributed Databases

- Definition of Distributed Databases
- DDBMS ,its characteristics
- Topology of DDBMS
- Advantages and Disadvantages of DDBMS
- Heterogeneous and Homogeneous Databases
- Distributed Data Storage
- Architecture of DDBMS
- DDBMS Design

Distributed Databases

"A logically interrelated collection of shared data physically distributed over a network."

DDBMS

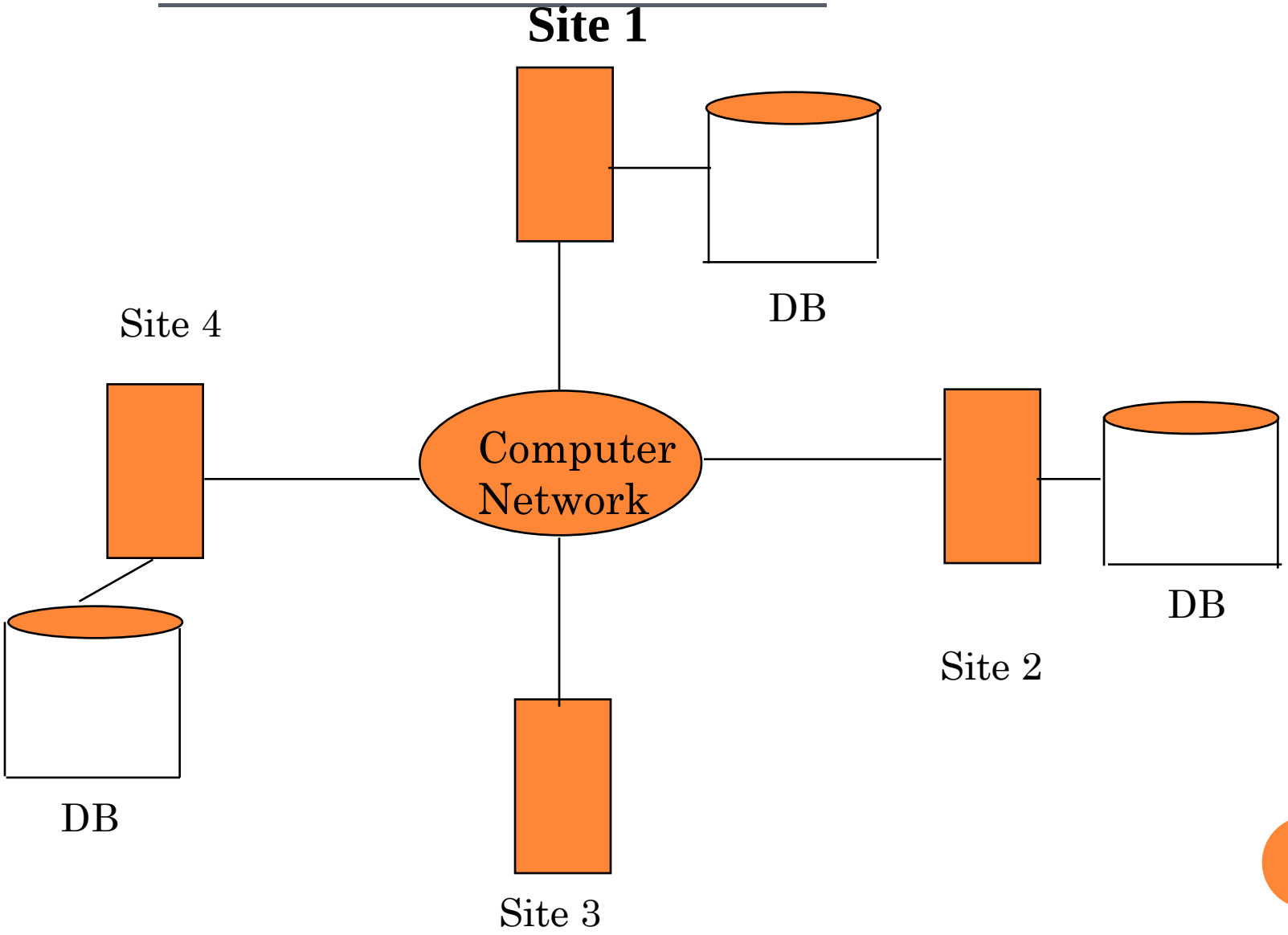
- **DDBMS stands for Distributed Database Management System.**

Characteristics of DDBMS

- **A DDBMS has the following characteristics**
  - **A collection of logically shared data**
  - **The data is split into a number of fragments**
  - **Fragments may be replicated**
  - **Fragments/Replicas are allocated to sites**
  - **The sites are linked by a communication network**
  - **The data at each site is under the control of a DBMS**
  - **The DBMS at each site can handle local applications autonomously**

- **NOTE:-It is not necessary for every site in the system to have its own local database.**

Topology of DDBMS



Advantages of DDBMS

- Reflects organizational structure
- Improved share ability and local autonomy
- Improved availability
- Improved reliability
- Improved Performance
- Economics
- Modular growth

Disadvantages of DDBMS

- Complexity
- Cost
- Security
- Integrity Control more difficult
- Lack of standards
- Lack of experience

Homogeneous Distributed Databases

- In a homogeneous distributed database
  - All sites have identical software
  - Are aware of each other and agree to cooperate in processing user requests.
  - Each site surrenders part of its autonomy in terms of right to change schemas or software
  - Appears to user as a single system

Heterogeneous Distributed Databases

- In a heterogeneous distributed database
  - Different sites may use different schemas and software
    - Difference in schema is a major problem for query processing
    - Difference in software is a major problem for transaction processing
  - Sites may not be aware of each other and may provide only
    limited facilities for cooperation in transaction processing

Distributed Data Storage

- Replication
  - System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.
- Fragmentation
  - Relation is partitioned into several fragments stored in distinct sites
- Replication and fragmentation can be combined
  - Relation is partitioned into several fragments: system maintains several identical replicas of each such fragment.

Data Replication

- A relation or fragment of a relation is replicated if it is stored redundantly in two or more sites.
- Full replication of a relation is the case where the relation is stored at all sites.
- Fully redundant databases are those in which every site contains a copy of the entire database.

Data Replication (Cont.)

- Advantages of Replication
  - Availability: failure of site containing relation *r* does not result in unavailability of *r* is replicas exist.
  - Parallelism: queries on *r* may be processed by several nodes in parallel.
  - Reduced data transfer: relation *r* is available locally at each site containing a replica of *r*.

- Disadvantages of Replication
  - Increased cost of updates: each replica of relation *r* must be updated.
  - Increased complexity of concurrency control: concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented.
    - One solution: choose one copy as primary copy and apply concurrency control operations on primary copy

Data Fragmentation

- Division of relation r into fragments r_1, r_2, …, r_n which contain sufficient information to reconstruct relation r.
- Horizontal fragmentation: each tuple of *r* is assigned to one or more fragments
- Vertical fragmentation: the schema for relation *r* is split into several smaller schemas
  - All schemas must contain a common candidate key (or superkey) to ensure lossless join property.
  - A special attribute, the tuple-id attribute may be added to each schema to serve as a candidate key.

- **Example : relation account with following schema**
- ***Account-schema* = (*branch-name*, *account-number*, *balance*)**

Horizontal Fragmentation of *account* Relation

branch-name	***acc-number***	***balance***
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62

$$account_1 = \sigma_{branch\text{-}name = \text{“Hillside”}}(account)$$

branch-name	*account-number*	*balance*
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

$$account_2 = \sigma_{branch\text{-}name = \text{“Valley view”}}(account)$$

Vertical Fragmentation of *employee-info* Relation

branch-name	*customer-name*	*tuple-id*
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5
Valleyview	Kahn	6
Valleyview	Green	7

$$deposit_1 = \Pi_{branch\text{-}name,\ customer\text{-}name,\ tuple\text{-}id}\ (employee\text{-}info)$$

Acc-number	*balance*	*tuple-id*
A-305	500	1
A-226	336	2
A-177	205	3
A-402	10000	4
A-155	62	5
A-408	1123	6
A-639	750	7

$$deposit_2 = \Pi_{account\text{-}number,\ balance,\ tuple\text{-}id}\ (employee\text{-}info)$$

Advantages of Fragmentation

- **Horizontal:**
  - **allows parallel processing on fragments of a relation**
  - **allows a relation to be split so that tuples are located where they are most frequently accessed**
- **Vertical:**
  - **allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed**
  - **tuple-id attribute allows efficient joining of vertical fragments**
  - **allows parallel processing on a relation**
- **Vertical and horizontal fragmentation can be mixed.**
  - **Fragments may be successively fragmented to an arbitrary depth.**

Data Transparency

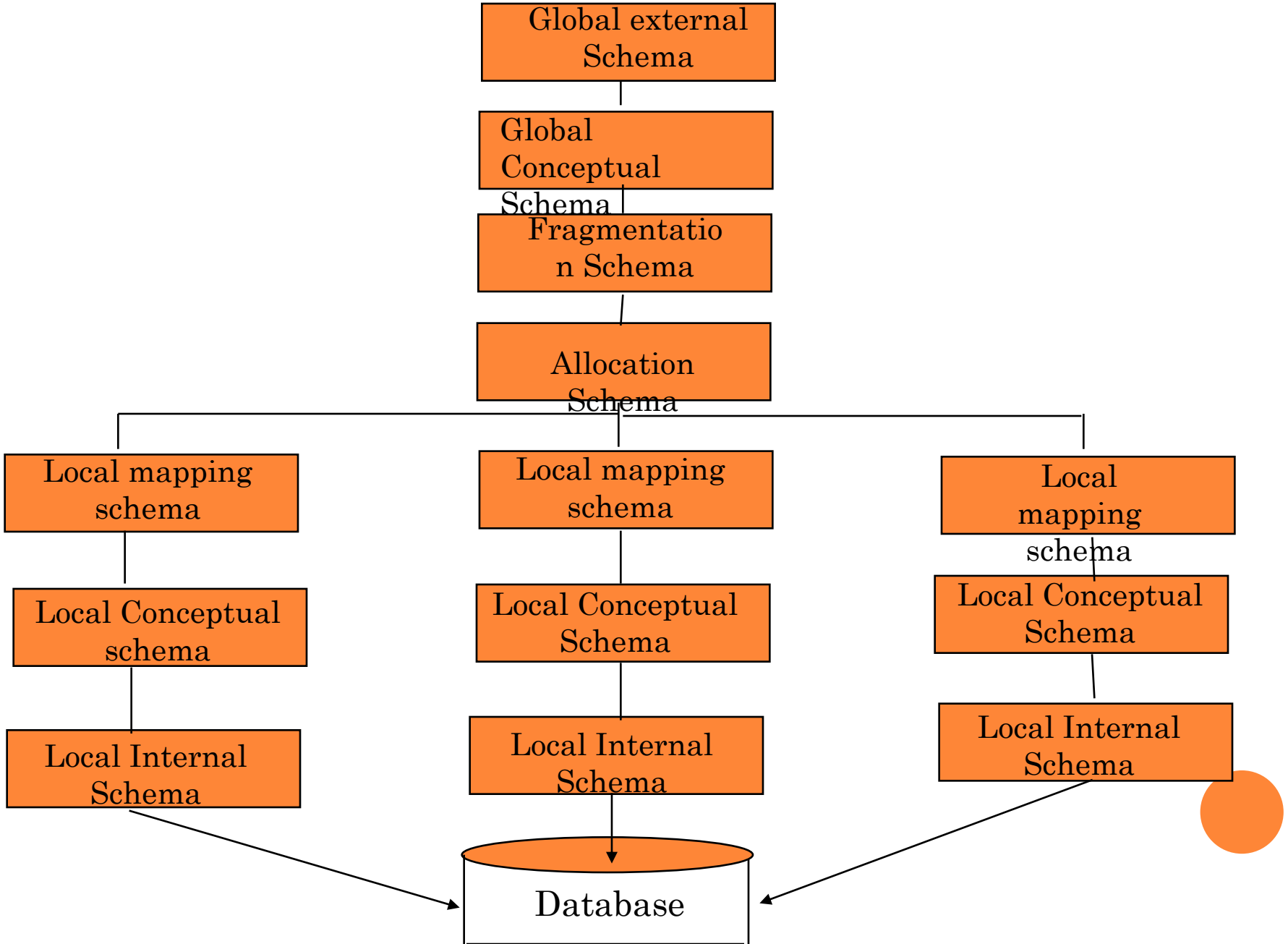
- Data transparency: Degree to which system user may remain unaware of the details of how and where the data items are stored in a distributed system
- Consider transparency issues in relation to:
  - Fragmentation transparency
  - Replication transparency
  - Location transparency

Architecture of DDBMS

- The reference architecture of DDBMS includes
  - A set of global external schema
  - A global conceptual schema
  - A fragmentation schema and allocation schema
  - A set of schemas for each local DBMS
- Global Conceptual Schema
  - Is a logical description of the whole database (as if it were not distributed)
  - Contains entities, relationships, constraints, security and integrity information.

- Fragmentation and Allocation schemas
  - fragmentation is how data is logically partitioned.
  - Allocation schema is where data is to be located
- Local Schemas
  - each local schema has its own set of schemas.

DDBMS


Global external Schema
Global Conceptual Schema
Fragmentatio n Schema
Allocation Schema
Local mapping schema
Local mapping schema
Local mapping schema
Local Conceptual schema
Local Conceptual Schema
Local Conceptual Schema
Local Internal Schema
Local Internal Schema
Local Internal Schema
Database

Distributed Database Design

- The factors to be considered for the design are:
  - Fragmentation
    - Horizontal Fragmentation
    - Vertical Fragmentation
  - Allocation
  - Replication

- Design should meet the following objectives
  - Locality of reference
  - Improved reliability and availability
  - Acceptable performance
  - Balanced storage capacities and costs
  - Minimal Communication costs