Distributed Databases

Distributed Databases

Introduction

- For proper functioning of any organization, there's a need for a well-maintained database.
- In the recent past, databases used to be centralized in nature.
- However, with the increase in globalization, organizations tend to be diversified across the globe.
- They may choose to distribute data over local servers instead of a central database.
- Thus, arrived the concept of **Distributed Databases**

Why Distributed Databases?

Factors that lead to distributed databases

- **Distributed Nature of Organizational Units**
  - Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe.
  - Each unit requires its own set of local data.
  - Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data**
  - The multiple organizational units often need to communicate with each other and share their data and resources.
  - This demands common databases or replicated databases that should be used in a synchronized manner.

Why Distributed Databases?

Factors that lead to distributed databases

- **Support for Both OLTP and OLAP**
  - Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data.
  - Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery**
  - One of the common techniques used in DDBMS is replication of data across different sites.
  - Replication of data automatically helps in data recovery if database in any site is damaged.
  - Users can access data from other sites while the damaged site is being reconstructed.
  - Thus, database failure may become almost inconspicuous to users.

Why Distributed Databases?

Factors that lead to distributed databases

- **Support for Multiple Application Software**
  - Most organizations use a variety of application software each with its specific database support.
  - DDBMS provides a uniform functionality for using the same data among different platforms.

Distributed Databases

Definition

- A distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.
  - A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users.

Distributed Databases

Features

1. Databases in the collection are logically interrelated with each other.
   - Often they represent a single logical database.
2. Data is physically stored across multiple sites.
   - Data in each site can be managed by a DBMS independent of the other sites.
3. The processors in the sites are connected via a network.
   - They do not have any multiprocessor configuration.
4. A distributed database is not a loosely connected file system.
5. A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system

Distributed Database Management System

Definition

- A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Distributed Database Management System

Features of DDBMS

1. It is used to create, retrieve, update and delete distributed databases.
2. It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
3. It ensures that the data modified at any site is universally updated.
4. It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
5. It is designed for heterogeneous database platforms.
6. It maintains confidentiality and data integrity of the databases.

Distributed Databases

Advantages of Distributed over centralized Databases

- **Modular Development**
  - If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning.
  - However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.
- **More Reliable**
  - In case of database failures, the total system of centralized databases comes to a halt.
  - However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance.
  - Hence DDBMS is more reliable.

Distributed Databases

Advantages of Distributed over centralized Databases

- **Better Response**
  - If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response.
  - On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.
- **Lower Communication Cost**
  - In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized.
  - This is not feasible in centralized systems.
- **Reflects organizational structure**
  - Fits the structure were many organizations are naturally distributed over several location.

Distributed Databases

Disadvantages of Distributed Databases

- **Need for complex and expensive software**
  - DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead**
  - Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity**
  - The need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution**
  - Responsiveness of queries is largely dependent upon proper data distribution.
  - Improper data distribution often leads to very slow response to user requests

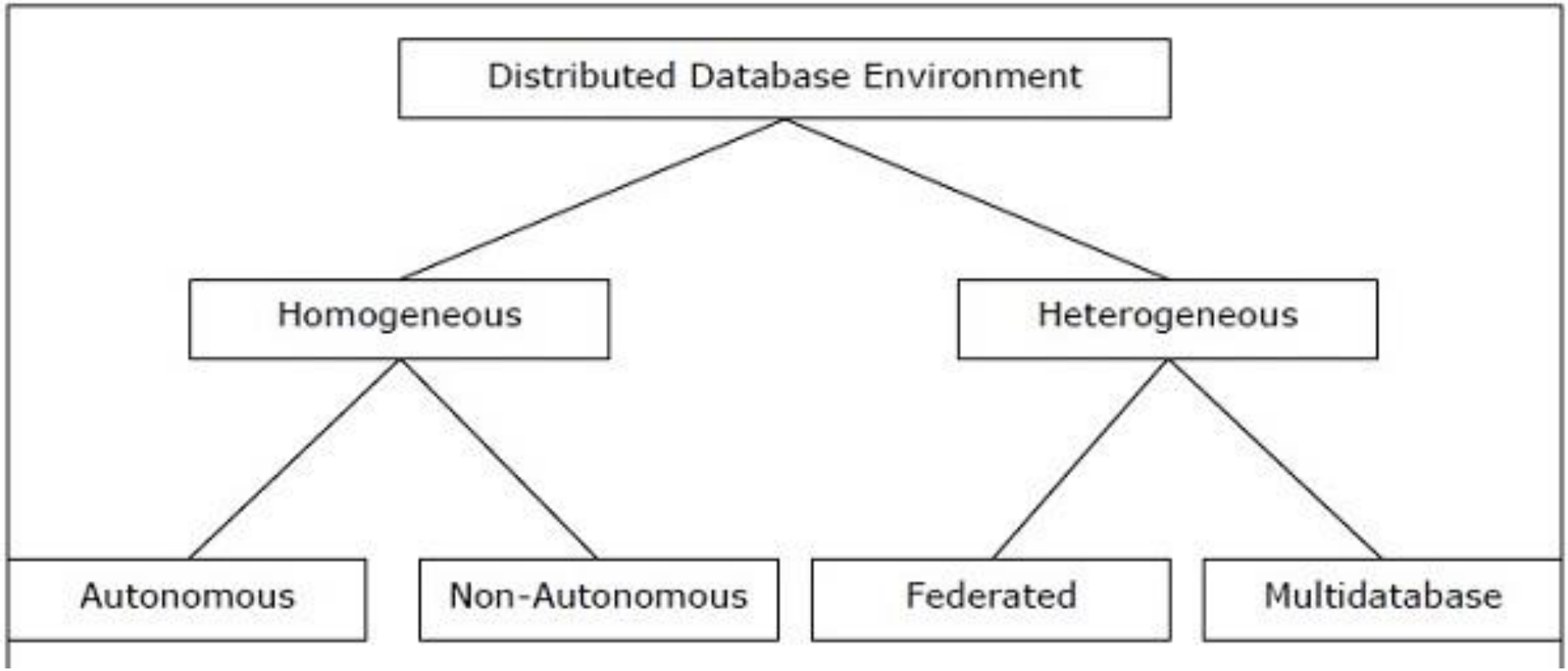
Applications of DDB

- Manufacturing - especially multi-plant manufacturing
- Military command and control
- Banking
- Corporate MIS
- Airlines
- Hotel chains
- Any organization which has a decentralized organization structure

Distributed Databases

Types of Distributed Databases

- Distributed databases are broadly classified into:
  1. homogeneous distributed database
  2. heterogeneous distributed database
  - Each has further sub-divisions as shown in the following



Distributed Databases

Homogeneous Distributed Databases

- In a homogeneous distributed database, all the sites use identical DBMS and operating systems.
- Its properties are:
  1. The sites use very similar software.
  2. The sites use identical DBMS or DBMS from the same vendor.
  3. Each site is aware of all other sites and cooperates with other sites to process user requests.
  4. The database is accessed through a single interface as if it is a single database.

Distributed Databases

Homogeneous Distributed Databases

Types of Homogeneous Distributed Database

•There are two types of homogeneous distributed database:

1.Autonomous

- Each database is independent and functions on its own.
- They are integrated by a controlling application and use message passing to share data updates.

2.Non-autonomous

- Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Distributed Databases

Heterogeneous Distributed Databases

- In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models.
- Its properties are:
  1. Different sites use dissimilar schemas and software.
  2. The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
  3. Query processing is complex due to dissimilar schemas.
  4. Transaction processing is complex due to dissimilar software.
  5. A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Distributed Databases

Heterogeneous Distributed Databases

Types of Heterogeneous Distributed Databases

1.Federated

- The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.

2.Un-federated

- The database systems employ a central coordinating module through which the databases are accessed.

Functions of DDBMS

In addition to having the functionality of a DBMS, DDBMS have other functionalities as follows:

i.Extended communication services to provide access to remote sites and allow transfer of queries and data among sites using a network.

ii.Extended system catalog to store data distribution details

iii.Distributed query processing, including query optimization and remote data access

iv.Extended security control to maintain appropriate authorization/access privileges to the distributed data.

v.Extended concurrence control to maintain consistency of replicated data.

vi.Extend recovery services to take account of failures of individual sites and the failures of communication links.

Key Design Issues

- Division and Location of Data
  - Data fragmentation
  - Replication
- Performance
- Transparency to the User
- Degree of homogeneity

Fragmentation

- Fragmentation independence refers to the ability of end users to store logically related information at different physical locations.
- There are two types of fragmentation independence:
  a. **Horizontal partitioning** ~ permits different rows of the same table to be stored at different remote sites.
     - This is commonly done by organizations that maintain several branch offices, each with an identical set of table structures.
  b. **Vertical partitioning** ~ refers to the ability of a distributed system to fragment information such that the data columns from the same logical tables are maintained across the network.

Horizontal Fragmentation

Projects with Budget less than/greater than or equal to 400,000

Pno	pname	budget	loc
H90	CAD/CAM	200000	Nairobi
S67	Database Dev	600000	H/Q
T67	Maintenance	450000	Kisumu
T90	Networks	300000	Mombasa
S45	School System	100000	Nairobi

Vertical Fragmentation

Info about project budgets/Info about project names and locations

Pno	pname	budget	loc
H90	CAD/CAM	200000	Nairobi
S67	Database Dev	600000	H/Q
T67	Maintenance	450000	Kisumu
T90	Networks	300000	Mombasa
S45	School System	100000	Nairobi

Fragmentation

Why fragment?

Usage:

- Apps work with views rather than entire relations.

Efficiency:

- Data stored close to where most frequently used.
- Data not needed by local applications is not stored.

Security:

- and so not available to unauthorized users.

Parallelism:

- With fragments as unit of distribution, it can be divided into several subqueries that operate on fragments

Replication

- Replication is the ability of a database to create copies of a master database at remote sites.
- These copies are sometimes called snapshots and may contain the whole database or any subcomponent of the database.
- For example, in a relational database, a CUSTOMER table may be snapped to many remote sites for read-only query.
  - Subsets of the CUSTOMER table may be specified, requesting only specific rows and columns, and these replications are refreshed on a periodic basis.
- Some of the data fragments may be duplicated and maintained in several sites.

Replication Alternatives

	Full Replication	Partial Replication	Partioned
Query Processing	Easy	Same Difficulty	Same Difficulty
Directory Management	Easy or Non-existent	Same Difficulty	Same Difficulty
Concurrency Control	Moderate	Difficulty	Easy
Reliability	Very High	High	Low
Realty	Possible Application	Realistic	Possible Application

Parallelism Requirements

- Have as much of the data required by *each* application at the site where the application executes
  - Full replication
- How about updates?
  - Updates to replicated data requires implementation of distributed concurrency control and commit protocols

System Expansion

- Issue is database scaling
- Emergence of microprocessor and workstation technologies
  - Client-server model of computing
- Data communication cost vs telecommunication cost

Distributed DBMS Issues

Distributed Database Design

- how to distribute the database
- replicated & non-replicated database distribution
- related problem in directory management

Query Processing

- convert user transactions to data manipulation instructions
- optimization problem

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

Transparency in a DDBMS

- Transparency hides implementation details from users.

- Overall objective: equivalence to user of DDBMs to centralised DBMS. FULL transparency not universally accepted objective

- Four main types:
  1. Distribution transparency
  2. Transaction transparency
  3. Performance transparency
  4. DBMS transparency (only applicable to heterogeneous)

Transparency in a DDBMS

Distribution Transparency

- Distribution transparency allows user to perceive database as single, logical entity.
- If DDBMS exhibits distribution transparency, user does not need to know:
  - fragmentation transparency: data is fragmented
  - Location transparency: location of data items
  - otherwise call this local mapping transparency
  - replication transparency: user unaware of replication of fragments

Transparency in a DDBMS

Transaction Transparency

- Transaction transparency ensures all distributed transactions maintain distributed database's integrity and consistency.

- Distributed transactions accesses data stored at more than one location.
- Each transaction is divided into no. of subtransactions, one for each site that has to be accessed.
- DDBMS must ensure the indivisibility of both the global transactions and each of the subtransactions.

Transparency in a DDBMS

Transaction Transparency

Concurrency transparency:

- All transactions must execute independently and be logically consistent with results obtained if transactions executed in some arbitrary serial order.
- Replication makes concurrency more complex

- Failure transparency:
  - Must ensure atomicity and durability of global transactions.
  - Means ensuring that subtransactions of global transaction either all commit or all abort.

Transparency in a DDBMS

Performance Transparency

- DDBMS: ~ no performance degradation due to distributed architecture.
- ~ determine most cost-effective strategy to execute a request.

- Distributed Query Processor (DQP) maps data request into ordered sequence of operations on local databases.
- ~ Must consider fragmentation, replication, and allocation schemas.
- DQP has to decide:
  - which fragment to access
  - which copy of a fragment to use
  - which location to use.
- ~ produces execution strategy optimized with respect to some cost function.
- Typically, costs associated with a distributed request include: I/O cost;
- CPU cost, communication cost.

Characteristics of DDBMS

1. It's a collection of logically related shared data
2. Data is split into a number of fragments
3. Fragments may be replicated
4. Fragments/replicas are allocated to sites
5. The Sites are linked to communication networks
6. The data at each site is under the control of a DBMS
7. The DBMS at each site can handle local applications, autonomously
8. Each DBMS participates in at least one global application.

Distributed Database Management System

Design Alternatives

- The distribution design alternatives for the tables in a DDBMS are as follows:
  1. Non-replicated and non-fragmented
  2. Fully replicated
  3. Partially replicated
  4. Fragmented
  5. Mixed

Distributed Database Management System

Design Alternatives

1. **Non-replicated & Non-fragmented**
   - In this design alternative, different tables are placed at different sites.
   - Data is placed so that it is at a close proximity to the site where it is used most.
   - It is most suitable for database systems where the percentage of queries needed to join information in tables placed at different sites is low.
   - If an appropriate distribution strategy is adopted, then this design alternative helps to reduce the communication cost during data processing.

Distributed Database Management System

Design Alternatives

2. **Fully Replicated**
   - In this design alternative, at each site, one copy of all the database tables is stored.
   - Since, each site has its own copy of the entire database, queries are very fast requiring negligible communication cost.
   - On the contrary, the massive redundancy in data requires huge cost during update operations.
   - Hence, this is suitable for systems where a large number of queries is required to be handled whereas the number of database updates is low.

Distributed Database Management System

Design Alternatives

3. **Partially Replicated**
   - Copies of tables or portions of tables are stored at different sites.
   - The distribution of the tables is done in accordance to the frequency of access.
   - This takes into consideration the fact that the frequency of accessing the tables vary considerably from site to site.
   - The number of copies of the tables (or portions) depends on how frequently the access queries execute and the site which generate the access queries.

Distributed Database Management System

Design Alternatives

4. **Fragmented**
   - In this design, a table is divided into two or more pieces referred to as fragments or partitions, and each fragment can be stored at different sites.
   - This considers the fact that it seldom happens that all data stored in a table is required at a given site.
   - Moreover, fragmentation increases parallelism and provides better disaster recovery.
   - Here, there is only one copy of each fragment in the system, i.e. no redundant data.
   - The three fragmentation techniques are:
     1. Vertical fragmentation
     2. Horizontal fragmentation
     3. Hybrid fragmentation

Distributed Database Management System

Design Alternatives

5. **Mixed Distribution**
   - This is a combination of fragmentation and partial replications.
   - Here, the tables are initially fragmented in any form (horizontal or vertical), and then these fragments are partially replicated across the different sites according to the frequency of accessing the fragments.

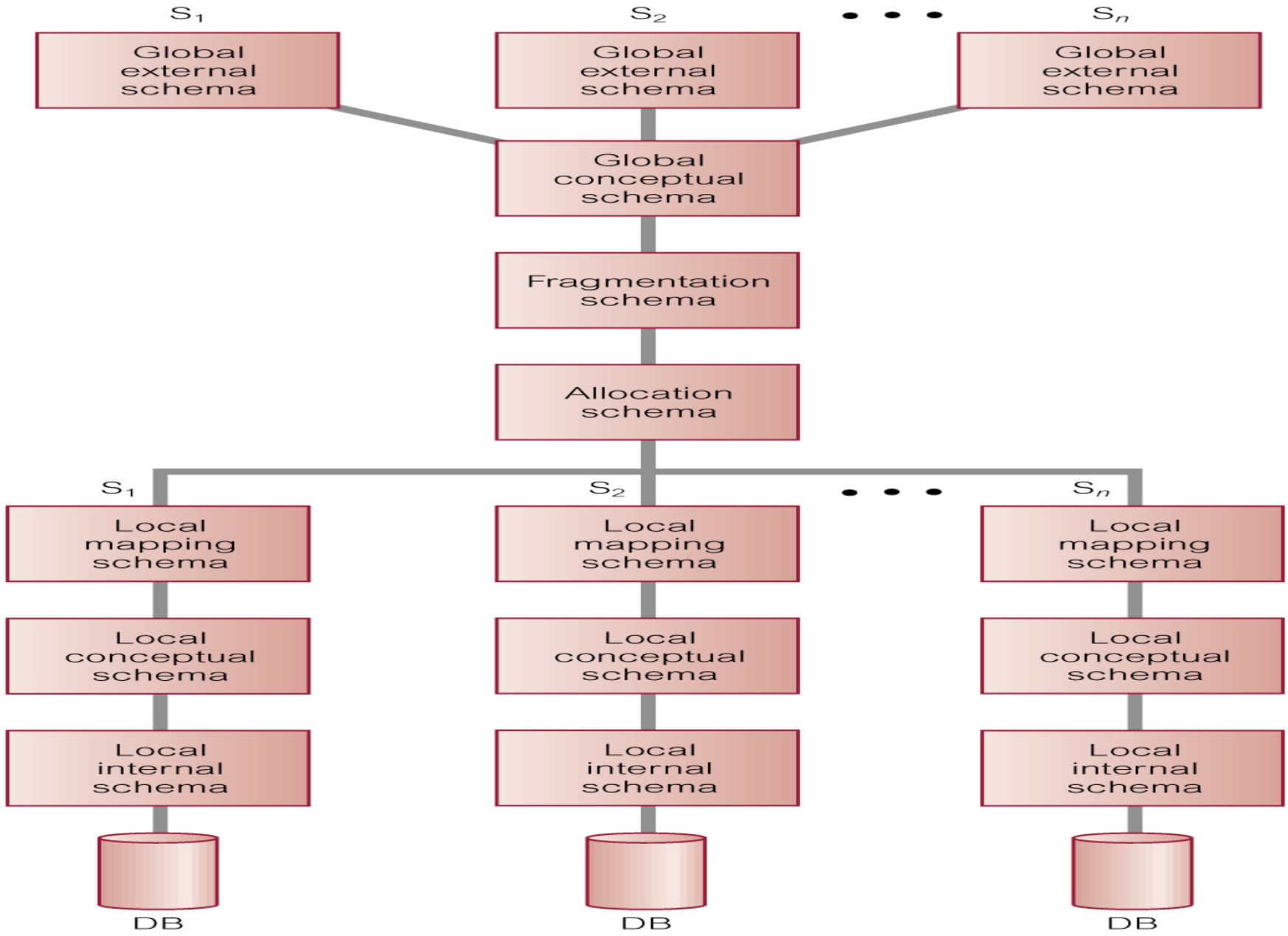
Architecture of a DDBMS

Owing to the diversity of distributed DBMSs, it is much more difficult to present an equivalent architecture that is generally applicable.

Our reference architecture consist of the following schemas:

- Global conceptual schema
  - This is a logical description of the whole database as if it were not distributed.
  - This level corresponds to the conceptual level of the ANSI-SPARC architecture and contains definitions of entities, relationships, constraints, security and integrity information.
  - It provides physical data independency from the distributed environment
  - The global external schemas provide logical data independency.

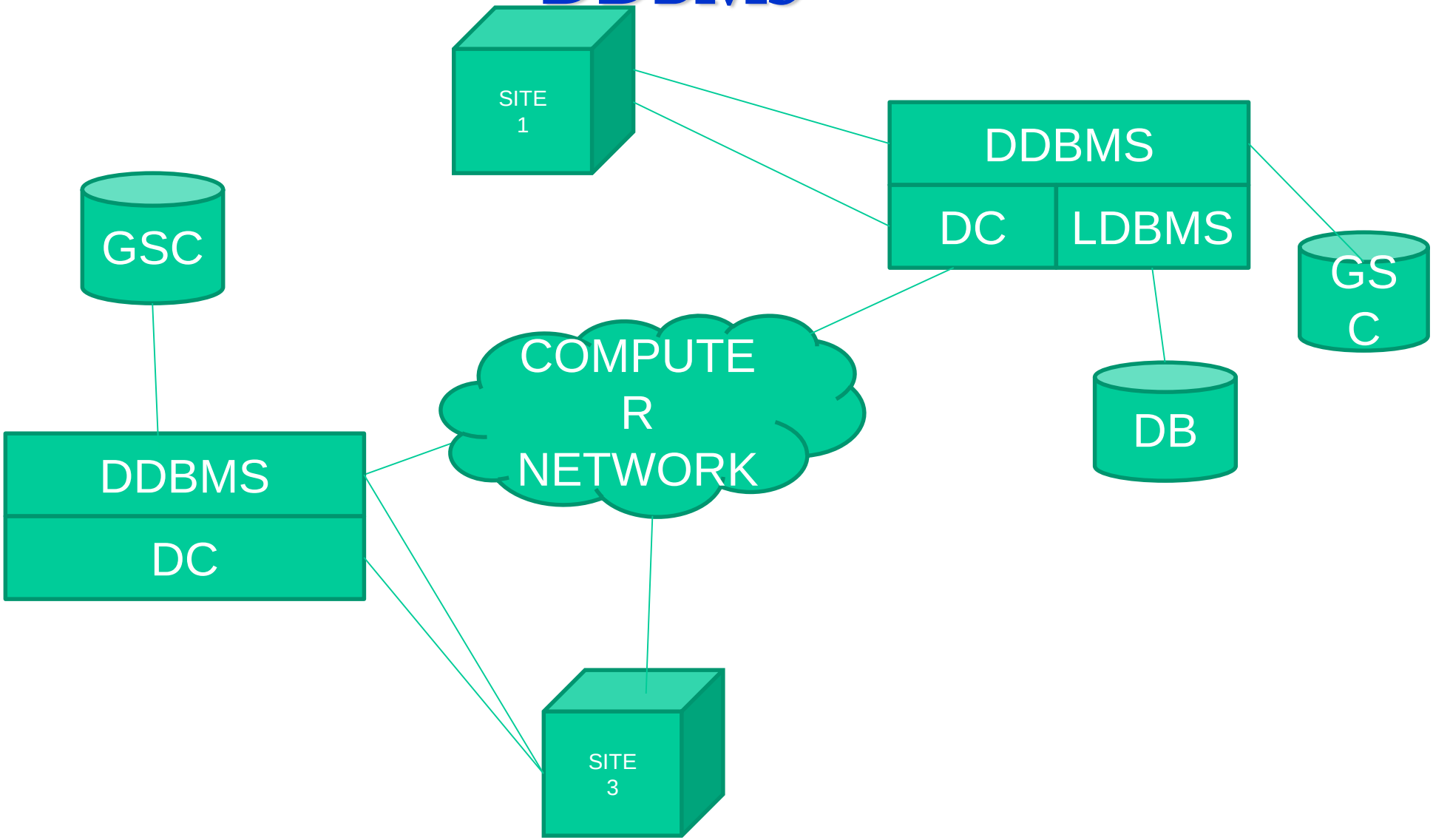
Reference Architecture for DDBMS



Architecture of a DDBMS

- Fragmentation and allocation schemas
  - Fragmentation schema is a description on how the data is to be logically partitioned.
  - The allocation schema is a description of where the data is to be located, taking account of any replication.
- Local Schemas
  - Each local DBMS has its own schemas
  - The local mapping schema maps fragments in the allocation schema into external objects in the local database.
  - It is DBMS independent and is the basis for supporting heterogeneous DBMSs

Component architecture for a DDBMS



Component architecture for a DDBMS

A component architecture for a DDBMS consist of four major components:

- Local DBMS component
  - A standard DBMS responsible for controlling the local data at each site that has a database
  - It has its own local system catalog that stores information about data held at that site
  - In homogenous system the LDBMS is i.e. is the same product replicated at each site though in heterogeneous system there would be at least two sites with different DBMS products.
- Data communication component
  - The software that enables all sites to communicate with each other
  - It contains information about the site and the links

Component architecture for a DDBMS

- Global system catalog
  - The GSC has the same functionality as the system catalog for a centralized system, i.e. holds information specific to the distributed nature of the system, such as the fragmentation, replication and allocation schemas
- Distributed DBMS component
  - It is the controlling unit for the entire system.
  - Performs all the functions of a DDBMS

Distributed Database Management System

DDBMS Architectures

DDBMS architectures are generally developed depending on three parameters :

1. **Distribution** – It states the physical distribution of data across the different sites.
2. **Autonomy** – It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
3. **Heterogeneity** – It refers to the uniformity or dissimilarity of the data models, system components and databases.

Distributed Database Management System

DDBMS Architectures

Some of the common architectural models are:

1. Client ~ Server Architecture for DDBMS
2. Peer ~ to ~ Peer Architecture for DDBMS
3. Multi ~ DBMS Architecture

•Each is discussed in the next slides

Distributed Database Management System

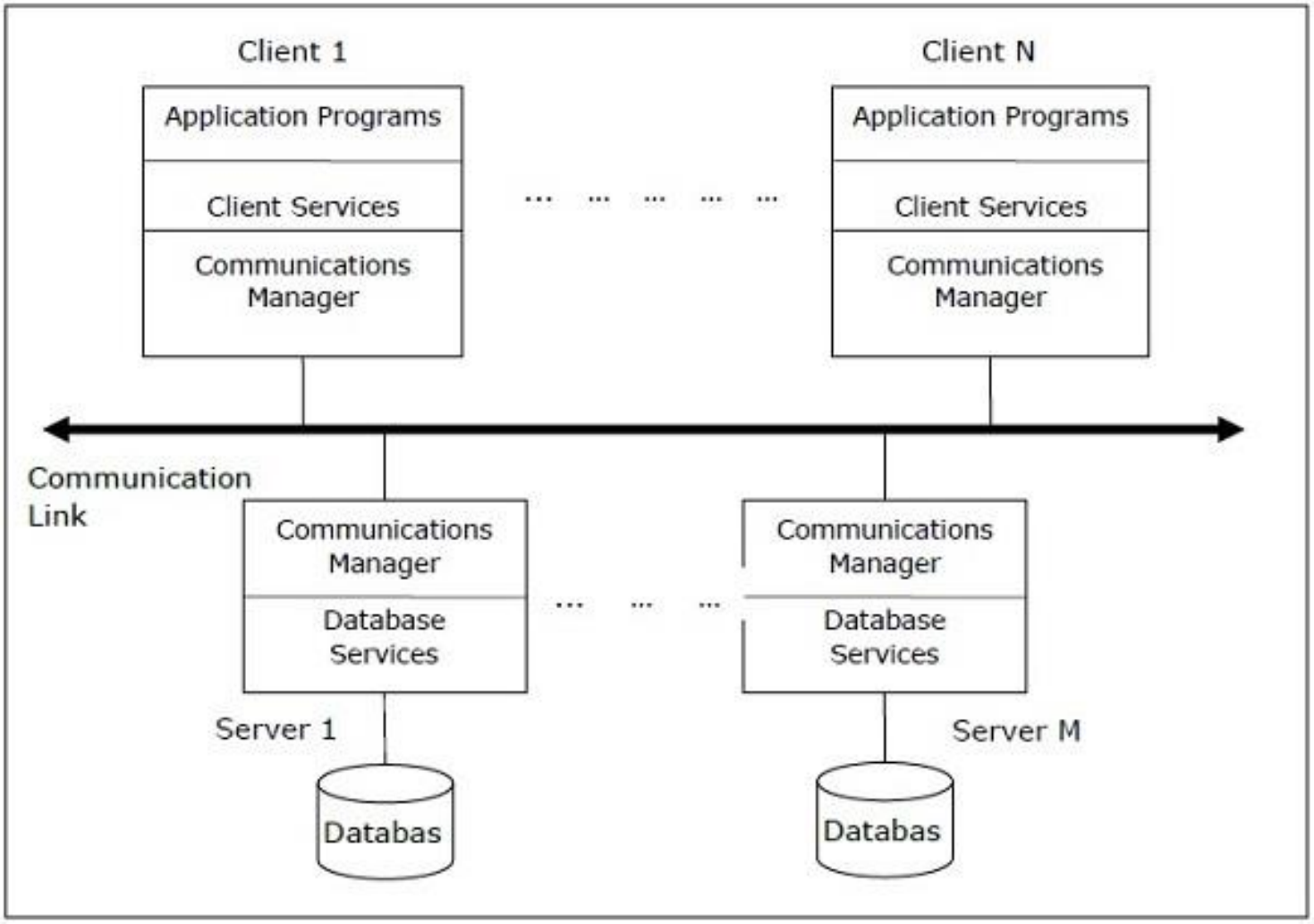
Client - Server Architecture for DDBMS

- This is a two-level architecture where the functionality is divided into servers and clients.
  - The server functions primarily encompass data management, query processing, optimization and transaction management.
  - Client functions include mainly user interface.
  - However, they have some functions like consistency checking and transaction management.
- The two different client - server architecture are:
  1. Single Server Multiple Client
  2. Multiple Server Multiple Client

Distributed Database Management System

Client ~ Server Architecture for DDBMS

Multiple Server Multiple Client



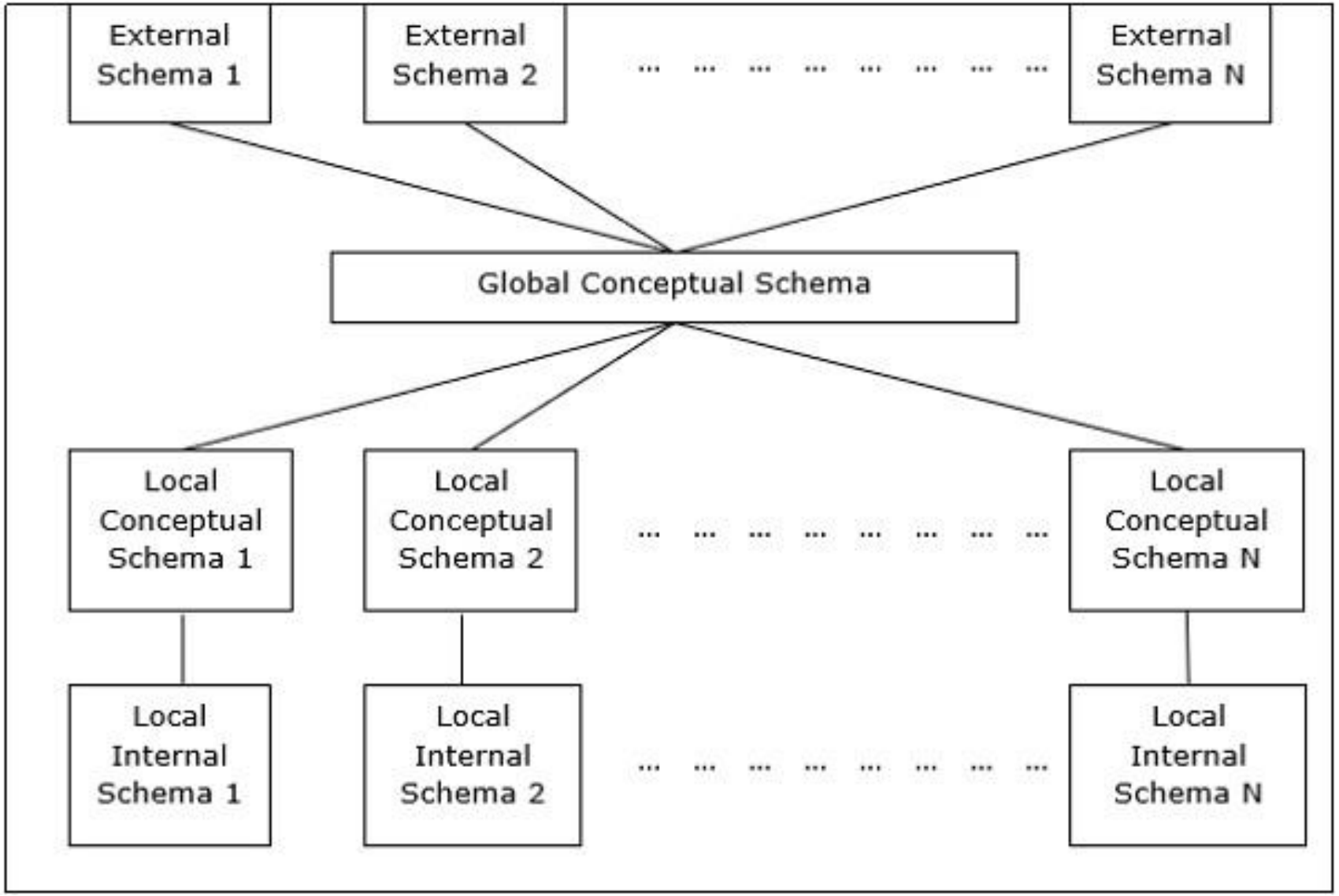
Distributed Database Management System

Peer~ to~Peer Architecture for DDBMS

- In these systems, each peer acts both as a client and a server for imparting database services.
- The peers share their resource with other peers and co~ordinate their activities.
- This architecture generally has four levels of schemas:
  1. **Global Conceptual Schema** – Depicts the global logical view of data.
  2. **Local Conceptual Schema** – Depicts logical data organization at each site.
  3. **Local Internal Schema** – Depicts physical data organization at each site.
  4. **External Schema** – Depicts user view of data.

Distributed Database Management System

Peer~ to~Peer Architecture for DDBMS



Distributed Database Management System

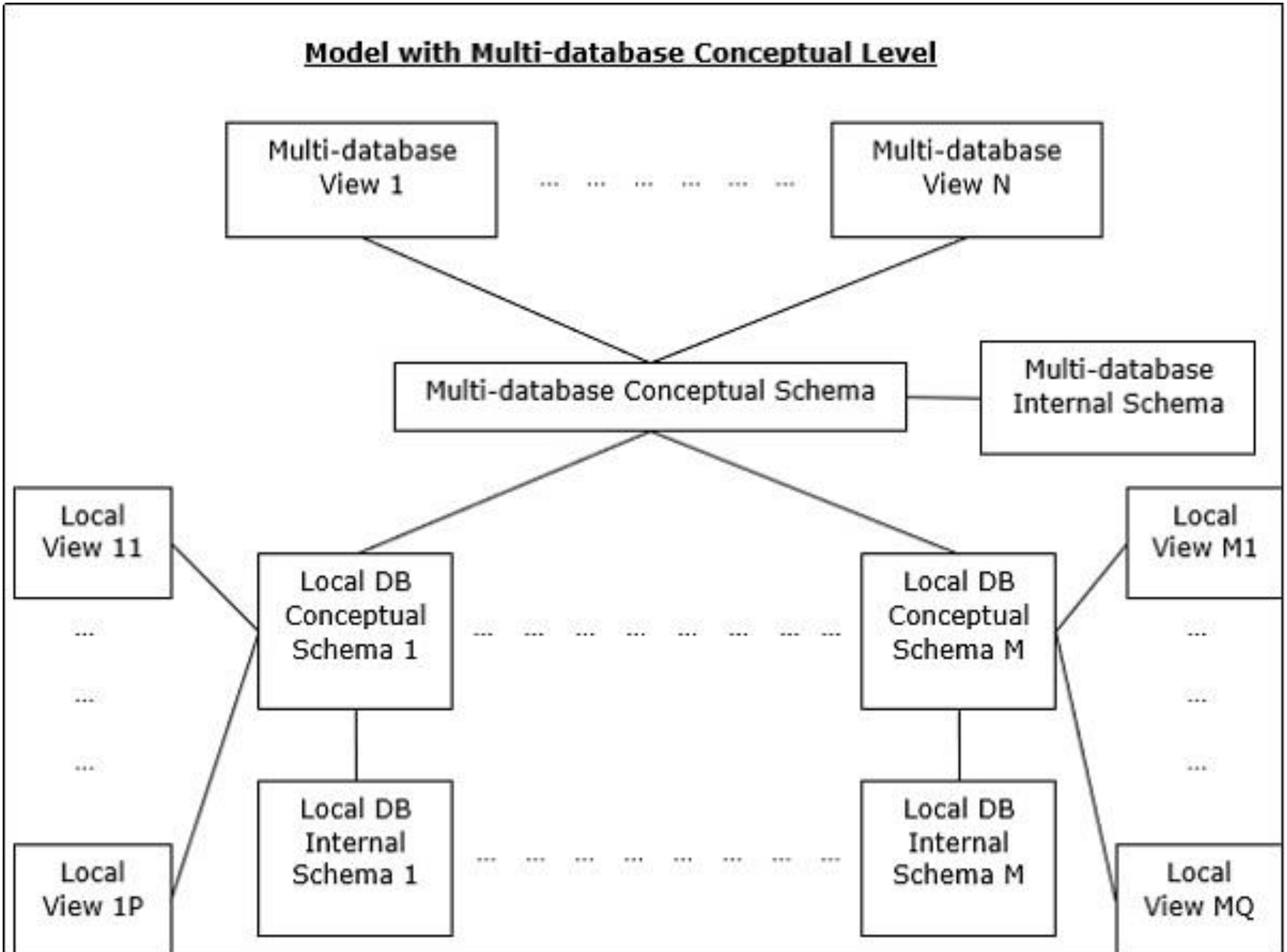
Multi - DBMS Architectures

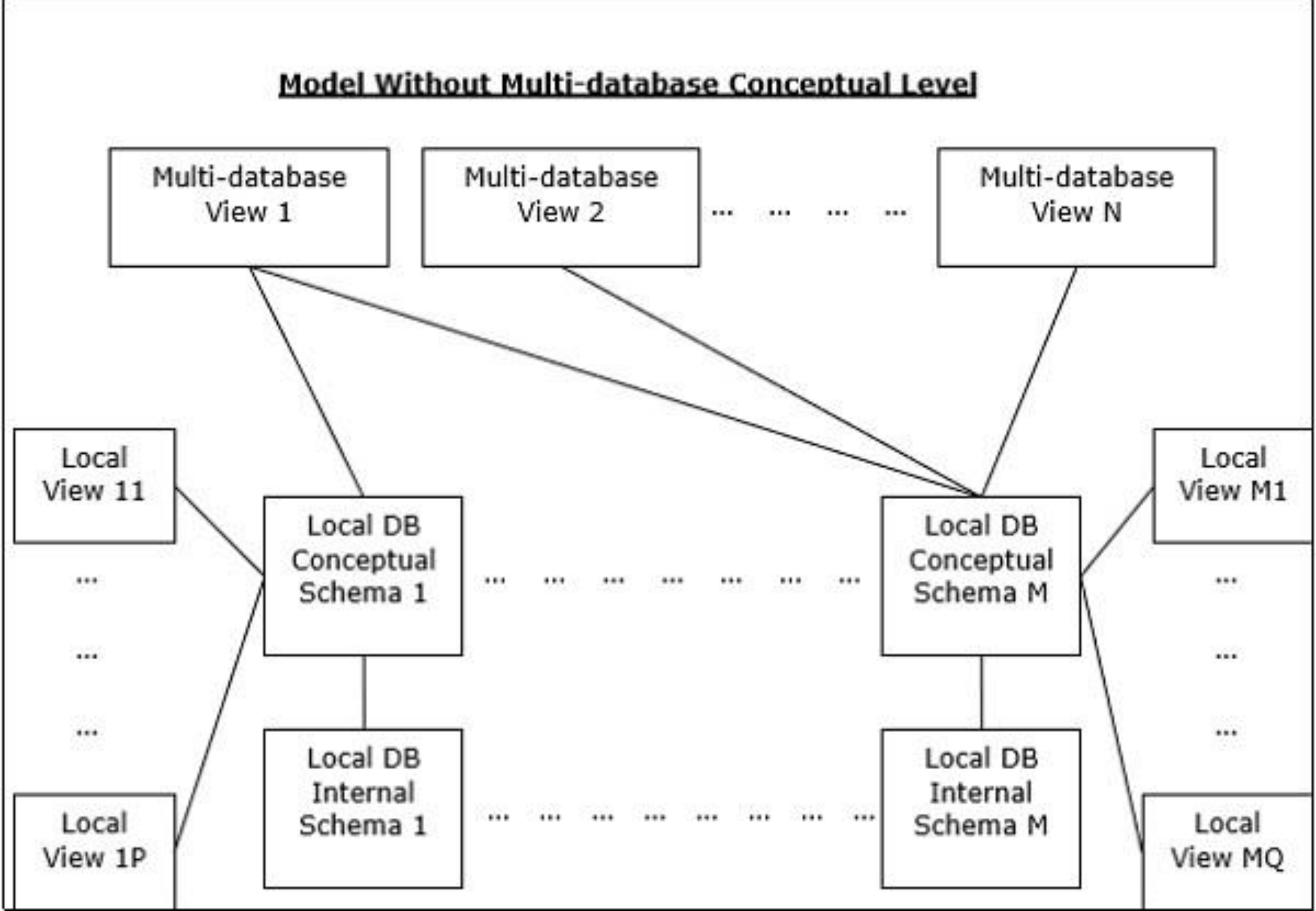
- This is an integrated database system formed by a collection of two or more autonomous database systems.
- Multi-DBMS can be expressed through six levels of schemas:
  1. **Multi-database View Level** – Depicts multiple user views comprising of subsets of the integrated distributed database.
  2. **Multi-database Conceptual Level** – Depicts integrated multi-database that comprises of global logical multi-database structure definitions.
  3. **Multi-database Internal Level** – Depicts the data distribution across different sites and multi-database to local data mapping.
  4. **Local database View Level** – Depicts public view of local data.
  5. **Local database Conceptual Level** – Depicts local data organization at each site.
  6. **Local database Internal Level** – Depicts physical data organization at each site.

Distributed Database Management System

Multi - DBMS Architectures

- There are two design alternatives for multi-DBMS:
  1. Model with multi-database conceptual level.
  2. Model without multi-database conceptual level.


Model with Multi-database Conceptual Level
Multi-database View 1
Multi-database View N
Multi-database Conceptual Schema
Multi-database Internal Schema
Local View 11
Local DB Conceptual Schema 1
Local DB Conceptual Schema M
Local View M1
Local View 1P
Local DB Internal Schema 1
Local DB Internal Schema M
Local View MQ


Model Without Multi-database Conceptual Level
Multi-database View 1
Multi-database View 2
Multi-database View N
Local View 11
Local DB Conceptual Schema 1
Local DB Conceptual Schema M
Local View M1
Local View 1P
Local DB Internal Schema 1
Local DB Internal Schema M
Local View MQ

Date's Twelve Rules for DDBMS[1]

Fundamental Principle

To the user, a distributed system should look exactly like a non-distributed system

1 Local Autonomy

- The sites in a distributed system should be autonomous.

  In this context, autonomy means that:

  - Local data is locally owned and managed;
  - Local operations remain purely local;
  - All operations at a given site are controlled by that site

Date's Twelve Rules for DDBMS2

2 No reliance on a Central Site

- There should be no one site without which the system cannot operate.
  - This implies that there should be no central servers for services such as transaction management, deadlock detection, query optimization, and management of the Global System Catalog

3 Continuous operation

- Ideally, there should never be a need for a planned system shutdown;

  for operations such as:
  - adding or removing a site from the system;
  - the dynamic creation and deletion of fragments at one or more sites

Date's Twelve Rules for DDBMS[3]

4 Location Independence (Transparency)

- The user should be able to access the database from any site. Furthermore, the user should be able to access all data as if it were stored at the user's site, no matter where it is physically stored

5 Fragmentation Independence

- The user should be able to access the data, no matter how it is fragmented.

Date's Twelve Rules for DDBMS[4]

6 Replication independence

- The user should be unaware that data has been replicated.
  - Thus, the user should not be able to access a particular copy of a data item directly, nor should the user have to specifically update all copies of a data item

7 Distributed query processing

- The system should be capable of processing queries that reference data at more than one site

Date's Twelve Rules for DDBMS[5]

8 Distributed Transaction Processing

- The system should support the transaction as the unit of recovery.
  - The system should ensure that both the global and local transactions conform to the ACID rules for transactions, namely: atomicity, consistency, isolation, and durability.

9 Hardware independence

- It should be possible to run the DDBMS on a variety of hardware platforms.

10 Operating system independence

- As a corollary to the previous rule, it should be possible to run the DDBMS on a variety of operating systems

Date's Twelve Rules for DDBMS[6]

11 Network Independence

- Again, it should be possible to run the DDBMS on a variety of disparate communication networks

12 Database Independence

- It should be possible to run different local DBMSs, perhaps supporting different underlying data models.
  - In other words, the system should support heterogeneity

References

- http://jcsites.juniata.edu/faculty/rhodes/dbms/distrib.htm
- https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_quick_guide.htm