Distributed database Management System

- A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

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

Distributed Database Management System

- 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.

- A Distributed Database Management System (DDBMS) consists of a single logical database that is split into a number of fragments.
- Each fragment is stored on one or more computers under the control of a separate DBMS, with the computers connected by a communications network.
- Each site is capable of independently processing user requests that require access to local data (that is, each site has some degree of local autonomy) and is also capable of processing data stored on other computers in the network.

- Users access the distributed database via applications.
- Applications are classified as those that do not require data from other sites (local Applications) and those that do require data from other sites (global applications). We require a DDBMS to have at least one global application.

Example of DDBMS

- Using distributed database technology, a bank may implement their database system on a number of separate computer systems rather than a single, centralized mainframe. The computer systems may be located at each local branch office: for example, Amritsar, Jalandhar, and Qadian. A network linking the computer will enable the branches to communicate with each other, and DDBMS will enable them to access data stored at another branch office. Thus a client living in Amritsar can also check his/her account during the stay in Jalandhar or Qadian.
- From the defination of the DDBMS, the system is expected to make the distribution transparent (invisible) to the user. Thus, the fact that a distributed database is split into fragments that can be stored on different computers and perhaps replicated should be hidden from the user. The objective of transparency is to make the distributed system appear like a centralized system. This is sometimes referred to as the fundamental principle of distributed DBMSs. This requirement provides significant functionality for the end-user but, unfortunately, creates many additional problems that have to be handled by the DDBMS.

- A DDBMS, therefore, has the following characteristics:
- a collection of logically related shared data
- n 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 communications network
- the data at each site is under the control of a DBMS
- the DBMS at each site can handle local applications, autonomously
- each DBMS participates in at least one global application

Distributed database	Centralized database
Consists of multiple database files located at different sites	Consists of a single, central database file
Allows multiple users to access and manipulate data	Bottlenecks when multiple users access same file simultaneously
Files delivered quickly from location nearest the user	Files may take longer to deliver to users
If one site fails, data is retrievable	Single site means downtime in cases of system failures
Multiple files from dispersed databases must be synchronized	Simpler to update and manage data in single, central system

Advantage

1. Data are located near the greatest demand site. The data in a distributed database system are dispersed to match business requirements which reduce the cost of data access.

2. Faster data access. End users often work with only a locally stored subset of the company's data.

3. Faster data processing. A distributed database system spreads out the systems workload by processing data at several sites.

4. Growth facilitation. New sites can be added to the network without affecting the operations of other sites.

5. Improved communications. Because local sites are smaller and located closer to customers, local sites foster better communication among departments and between customers and company staff.

6. Reduced operating costs. It is more cost-effective to add workstations to a network than to update a mainframe system. Development work is done more cheaply and more quickly on low-cost PCs than on mainframes.

7. User-friendly interface. PCs and workstations are usually equipped with an easy-to-use graphical user interface (GUI). The GUI simplifies training and use for end users.

8. Less danger of a single-point failure. When one of the computers fails, the workload is pickedup by other workstations. Data are also distributed at multiple sites.

9. Processor independence. The end user is able to access any available copy of the data, and an end user's request is processed by any processor at the data location.

Advantages of Distributed Database

1. Increased reliability and availability
   - Even when a component fails the database may continue to function albeit at a reduced level
2. Allow Local control over data.
   - Local control promotes data integrity and administration
3. Modular growth
   - Easy to add a connection to a new location
   - Less chance of disrupting existing users during expansion
4. Lower communication costs.
5. Faster response for certain queries.
   - Query local data
   - Parallel queries

Disadvantage

1. **Complexity of management and control.** Applications must recognize data location, and they must be able to stitch together data from various sites. Database administrators must have the ability to coordinate database activities to prevent database degradation due to data anomalies.
2. **Technological difficulty**. Data integrity, transaction management, concurrency control, security, backup, recovery, query optimization, access path selection, and so on, must all be addressed and resolved.

3. **Security**. The probability of security lapses increases when data are located at multiple sites. The responsibility of data management will be shared by different people at several sites.

4. **Lack of standards**. There are no standard communication protocols at the database level. (Although TCP/IP is the de facto standard at the network level, there is no standard at the application level.) For example, different database vendors employ different—and often incompatible—techniques to manage the distribution of data and processing in a DDBMS environment.

5. **Increased storage and infrastructure requirements**. Multiple copies of data are required at different sites, thus requiring additional disk storage space.

6. **Increased training cost**. Training costs are generally higher in a distributed model than they would be in a centralized model, sometimes even to the extent of offsetting operational and hardware savings.

7. **Costs.** Distributed databases require duplicated infrastructure to operate (physical location, environment, personnel, software, licensing, etc.)

Disadvantages of Distributed Database

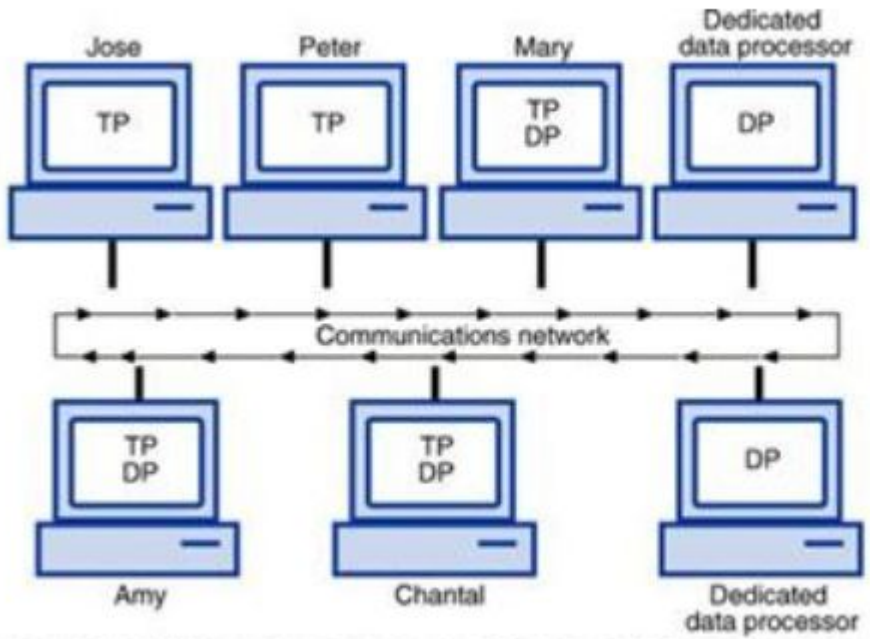
- Software cost and complexity.
- Processing overhead.
- Data integrity exposure.
- Slower response for certain queries.
  - If data are not distributed properly, according to their usage, or if queries are not formulated correctly, queries can be extremely slow

Components of DDBMS

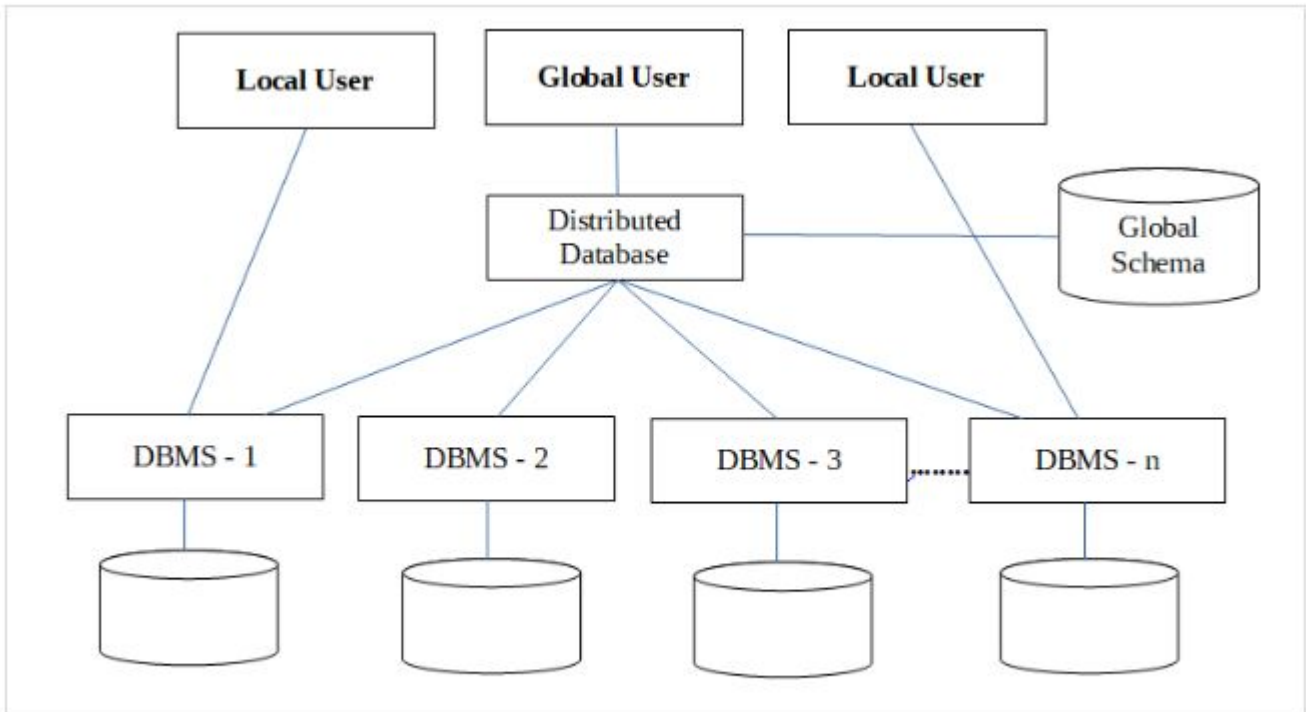
Hardware	Each processing site (or node) that forms the database system can consist of various types of hardware. Nodes can be mainframes, minicomputers or microcomputers. **Homogeneous** nodes combine the same type of hardware whereas **Heterogeneous** nodes combine a mixture of hardware.
Communications Media	**Network hardware and software** allows each node to communicate and exchange data with other nodes that comprise the network. Local area networks typically use cables to transmit data from node to node, whereas telephone lines or satellites are used for more widely dispersed sites.

Software	**A distributed database management system** is a collection of data processors and transaction processors.
Data Processors (DPs)	Are programs that store and retrieve data at local sites. A DP could be an independent database management system such as Access or Oracle, or it could be a subset of the distributed database management system.

Transaction processors (TPs)	Are programs that control and co-ordinate query and transaction data requests from local or remote sites. Data requests are analysed by the TP to determine update or retrieval locations required by the data request. TP's do this by accessing the **Distributed Data Catalog (DDC)** which contains a description of the entire database. Once specific data locations are determined, the TP then transfers the data requests to the appropriate data processors. A TP could already exist as part of the distributed database management system or it could be specifically written. TP features can also be manually incorporated into queries and transactions, and you will see examples of this when we explore distributed database transparency features in the next section.



Note: Each TP can access data on any DP, and each DP handles all requests for local



Let us now discuss them one by one −

DDBMS TYPES:

Homogeneous vs. Heterogeneous Distributed Database

- ***Homogeneous Distributed Database -***
  - The same DBMS is used at each node
  - Difficult for most organizations to force a homogeneous environment
- ***Heterogeneous Distributed Database***
  - Potentially different DBMS are used at each node
  - Much more difficult to manage

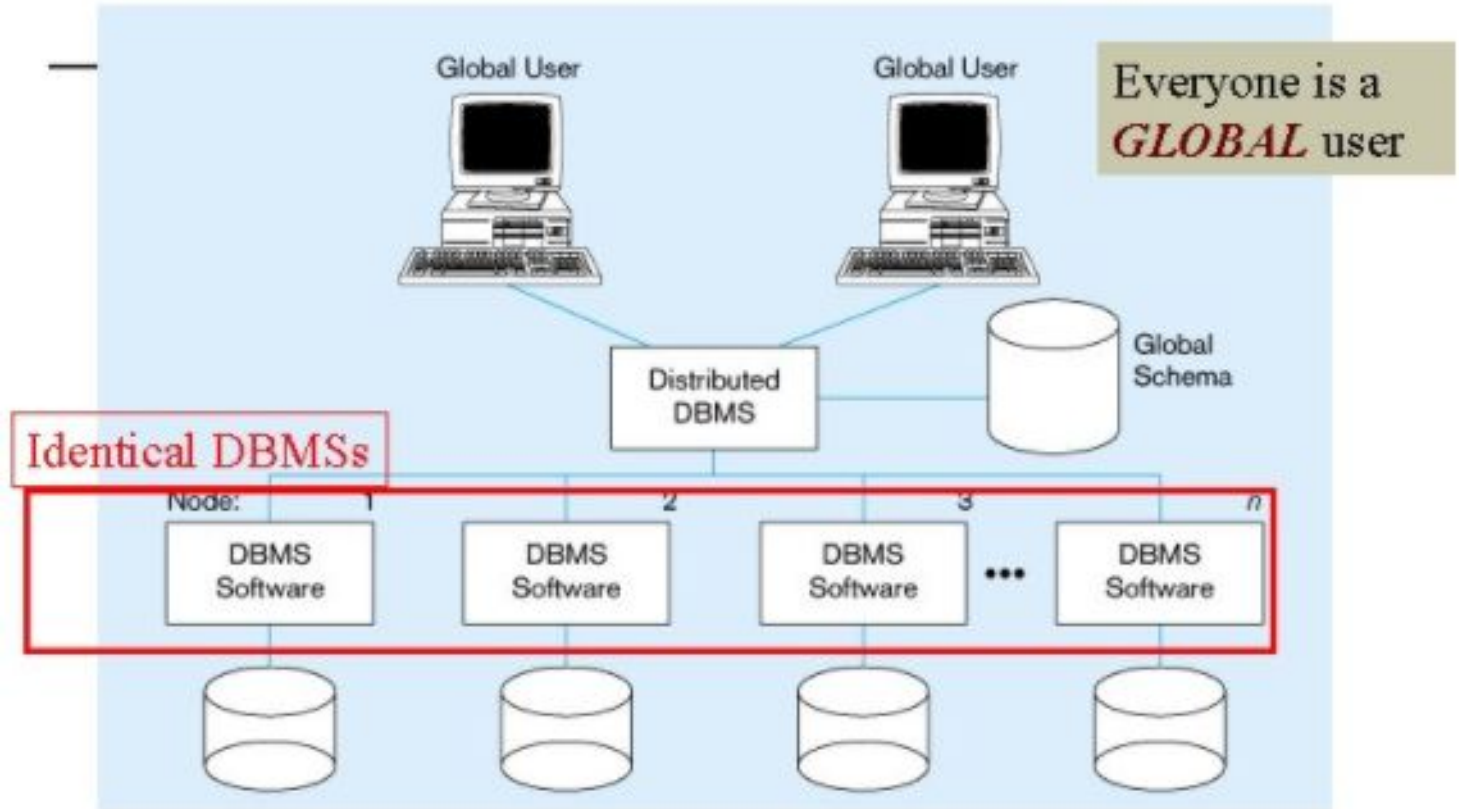
Typical Homogeneous Environment

- Data distributed across all the nodes.
- Same DBMS at each node.
- A central DBMS coordinates database access and update across the notes
  - No exclusively local data
- All access is through one, global schema.
  - The global schema is the *union* of all the local schema.

Homogeneous Database

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

Figure 13-2 – Homogeneous Database

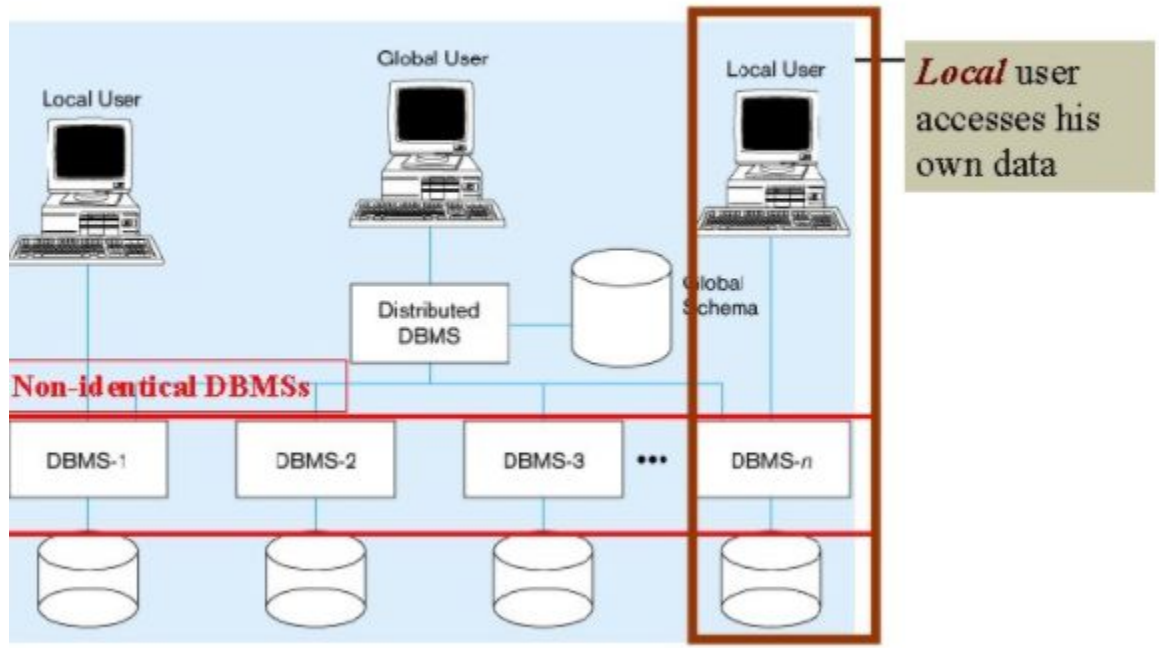


Typical Heterogeneous Environment

- 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.

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

Figure 13-3 –Typical Heterogeneous Environment



Local user accesses his own data

Fragmentation

- Fragmentation is a process of dividing the whole or full database into various subtables or sub relations so that data can be stored in different systems.
- The small pieces of sub relations or subtables are called *fragments*.
- These fragments are called logical data units and are stored at various sites. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn’t any loss of data).

- **Advantages :**
- As the data is stored close to the usage site, the efficiency of the database system will increase
- Local query optimization methods are sufficient for some queries as the data is available locally
- In order to maintain the security and privacy of the database system, fragmentation is advantageous
- **Disadvantages :**
- Access speeds may be very high if data from different fragments are needed
- If we are using recursive fragmentation, then it will be very expensive

- We have three methods for data fragmenting of a table:
- **Horizontal fragmentation**
- **Vertical fragmentation**
- **Mixed or Hybrid fragmentation**

- Horizontal fragmentation refers to the process of dividing a table horizontally by assigning each row or (a group of rows) of relation to one or more fragments.
- These fragments are then be assigned to different sides in the distributed system.
- Some of the rows or tuples of the table are placed in one system and the rest are placed in other systems.
- The rows that belong to the horizontal fragments are specified by a condition on one or more attributes of the relation. In relational algebra horizontal fragmentation on table T, can be represented as follows:

For example, consider an EMPLOYEE table (T) :

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

This EMPLOYEE table can be divided into different fragments like:

EMP 1 = $\sigma_{Dep=1}$ EMPLOYEE

EMP 2 = $\sigma_{Dep=2}$ EMPLOYEE

These two fragments are: T1 fragment of Dep = 1

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1

Similarly, the T2 fragment on the basis of Dep = 2 will be :

Eno	Ename	Design	Salary	Dep
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

Now, here it is possible to get back T as **T = T1 ∪ T2 ∪ .... ∪ TN**

Vertical Fragmentation

- Vertical fragmentation refers to the process of decomposing a table vertically by attributes are columns. In this fragmentation, some of the attributes are stored in one system and the rest are stored in other systems.
- This is because each site may not need all columns of a table. In order to take care of restoration, each fragment must contain the primary key field(s) in a table.
- The fragmentation should be in such a manner that we can rebuild a table from the fragment by taking the natural JOIN operation and to make it possible we need to include a special attribute called ***Tuple-id*** to the schema.
- For this purpose, a user can use any super key. And by this, the tuples or rows can be linked together. The projection is as follows:

For example, consider an EMPLOYEE table (T) :

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

For example, for the EMPLOYEE table we have T1 as :

Eno	Ename	Design	Tuple_id
101	A	abc	1
102	B	abc	2
103	C	abc	3
104	D	abc	4
105	E	abc	5

For the second. sub table of relation after vertical fragmentation is given as follows :

Salary	Dep	Tuple_id
3000	1	1
4000	2	2
5500	3	3
5000	1	4
2000	4	5

This is T2 and to get back to the original T, we join these two fragments T1 and T2 as $\pi_{EMPLOYEE}$ **(T1 ⋈ T2)**

- **Mixed Fragmentation**
- The combination of vertical fragmentation of a table followed by further horizontal fragmentation of some fragments is called **mixed or hybrid fragmentation**. For defining this type of fragmentation we use the SELECT and the PROJECT operations of relational algebra. In some situations, the horizontal and the vertical fragmentation isn't enough to distribute data for some applications and in that conditions, we need a fragmentation called a mixed fragmentation.
- Mixed fragmentation can be done in two different ways:

1. The first method is to first create a set or group of horizontal fragments and then create vertical fragments from one or more of the horizontal fragments.
2. The second method is to first create a set or group of vertical fragments and then create horizontal fragments from one or more of the vertical fragments. The original relation can be obtained by the combination of JOIN and UNION operations which is given as follows:

For example, for our **EMPLOYEE** table, below is the implementation of mixed fragmentation is $\pi_{Ename, Design}(\sigma_{Eno < 104}(EMPLOYEE))$

The result of this fragmentation is:

Ename	Design
A	abc
B	abc
C	abc