LECTURE SESSION SEVEN

DISTRIBUTED DATABASES

Learning Objectives

7.1 Introduction

7.2 learning outcomes

7.3 Motivation Towards Distributed Databases

7.4 Distributed databases

7.4.1 Features of a distributed database

7.5 Distributed Database Management System (DDBMS)

7.5.1 Features of a Distributed Database Management System (DDBMS)

7.5.2 Functions of DDBMS

7.5.3 Factors Encouraging DDBMS

7.6 Advantages of Distributed Databases

7.7 Adversities of Distributed Databases

7.8 Distributed DBMS - Database Environments

7.8.1 Homogeneous Distributed Databases

7.8.2 Heterogeneous Distributed Databases

7.9 Distributed DBMS Architectures

7.9.1 Architectural Models

7.10 Design Alternatives

7.11 Distributed DBMS - Design Strategies

7.11.1 Data Replication

7.11.2 Fragmentation

7.11.3 DDBMS - Distribution Transparency

7.12 Date's Twelve Rules for DDBMS

7.13 Summary

7.14 Review activity

7.15 References and Further Reading

7.1 Introduction

Hello everyone!!! Welcome to today's session, last week we concluded on how concurrency control techniques are used to control conflicts, deadlocks and starvation. In today's lecture session, we shall give an overview of databases and Database Management Systems (DBMS, factors that lead to distributed databases and the architecture of distributed databases systems. I hope you are going to enjoy, welcome.

7.2 Learning Outcomes

At the end of this lecture, you should be able to:

1. Explain the concept of distributed database systems and distributed database management systems.
2. Describe the essential characteristics of distributed database systems.
3. Discuss different types of distributed systems
4. Discuss the architecture of a distributed database management systems
5. Describe the various design issues for distributed databases.

7.3 Motivation Towards Distributed Databases

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. Companies may choose to distribute data over local servers instead of a central database. Thus, arrived the concept of Distributed Databases. The move towards distributed databases is fueled by the drastic advances in technology particularly the microprocessor technology which has made computational entities more and more powerful and cheap. Equally enough, high-speed computer networks have made interconnection of computational entities possible at a wide range of scales and speeds hence making development of distributed systems possible.

Distributed databases have of late become an integral part of business computing. They are preferably used by organizations that have numerous offices or storefronts in different geographical locations. Typically, an individual branch in a distributed database system interacts primarily with the data that pertain to its own operations, with a much less frequent need for general company data.

7.4 Distributed Database

A distributed database is a set of interconnected databases that is distributed over the computer network or internet.

Definition

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

7.4.1 Features of a distributed database

The features of a distributed database include:

- 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 is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

7.5 Distributed Database Management System (DDBMS)

In a distributed database, there are a number of databases that may be geographically distributed all over the world. A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users.

Definition

A Distributed Database Management System (DDBMS) manages the distributed database and provides mechanisms so as to make the databases transparent to the users

In distributed database systems, data is intentionally distributed among multiple nodes so that all computing resources of the organization can be optimally used.

7.5.1 Features of a Distributed Database Management System (DDBMS)

The features of a DDBMS include the following:

- 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 is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

7.5.2 Functions of DDBMS

A distributed database management systems is used to perform the following functions:

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

7.5.3 Factors Encouraging DDBMS

There are several factors contributed to the move from centralised (single-processor) systems to decentralised (multi-processor) and distributed systems. The most influential factors that encourage moving over to DDBMS include:

- **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.
- **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.

- **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.

7.6 Advantages of Distributed Databases

Following are the advantages of distributed databases 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.
- **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 where many organizations are naturally distributed over several location.

7.7 Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

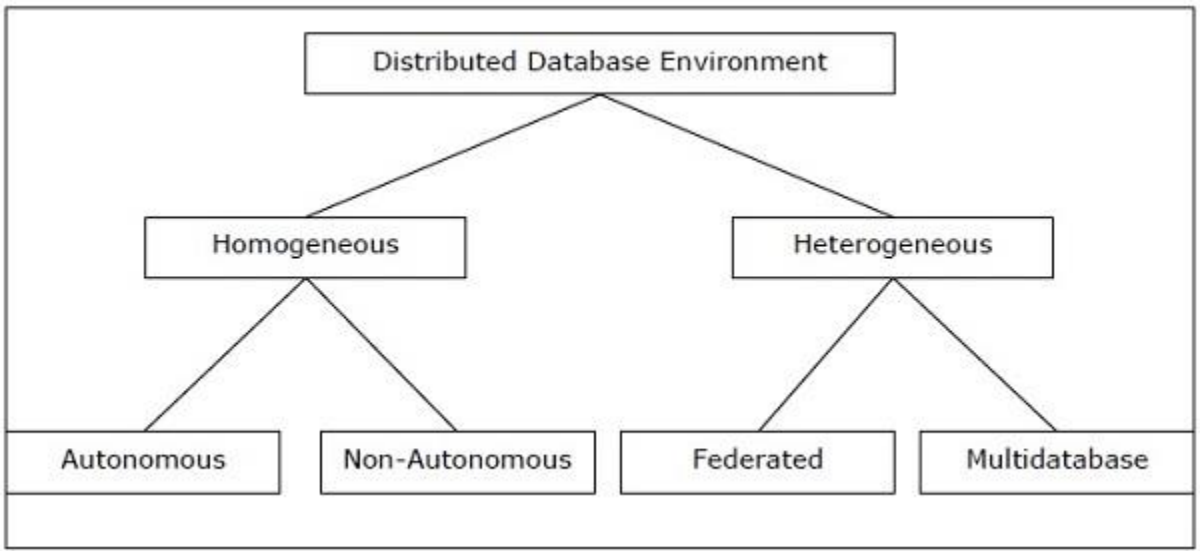
- **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide 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.

7.8 Distributed DBMS - Database Environments

Distributed databases can be classified into homogeneous and heterogeneous databases having further divisions as discussed below:

Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



7.8.1 Homogeneous Distributed Databases

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.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed 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.

7.8.2 Heterogeneous Distributed Databases

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.

Types of Heterogeneous Distributed Databases

- **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.

7.9 Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters −

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

7.9.1 Architectural Models

Some of the common architectural models are −

- Client - Server Architecture for DDBMS
- Peer - to - Peer Architecture for DDBMS
- Multi - DBMS Architecture

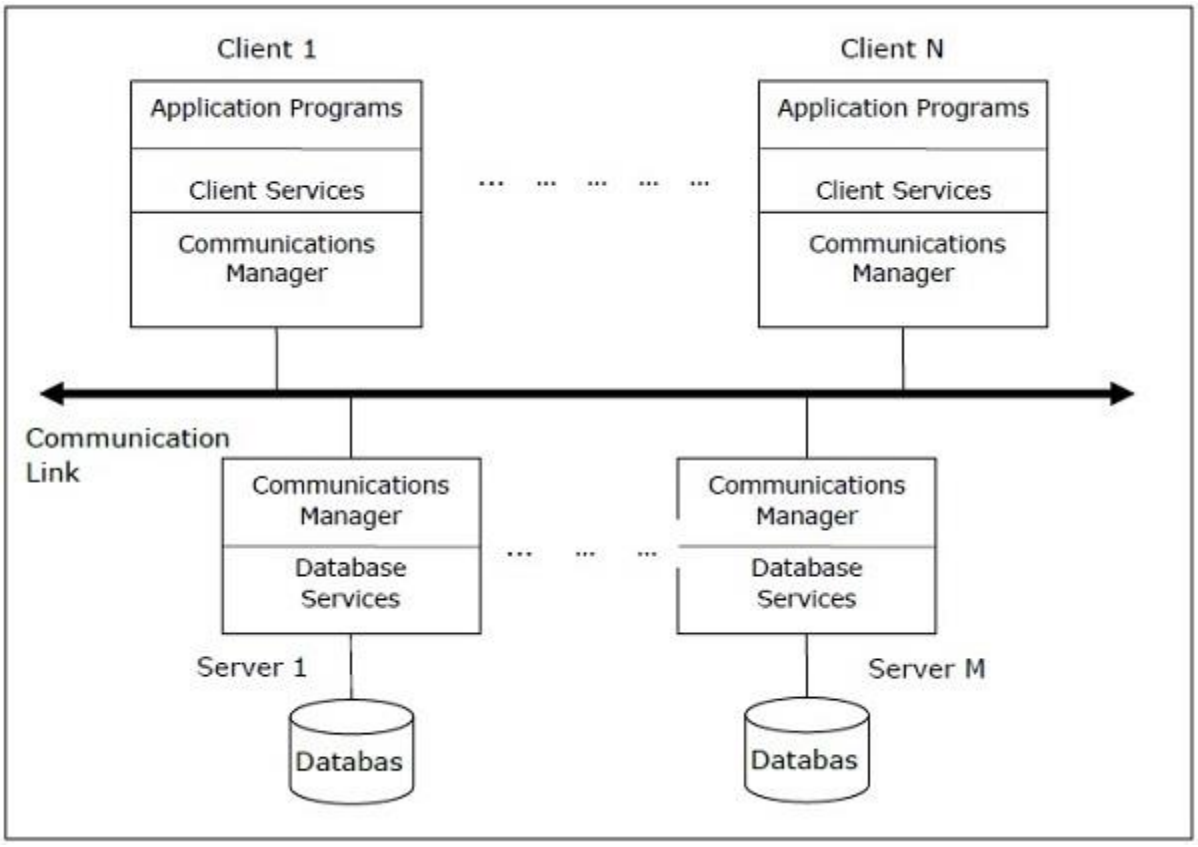
a. Client - Server Architecture for DDBMS

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 −

- Single Server Multiple Client
- Multiple Server Multiple Client (shown in the following diagram)

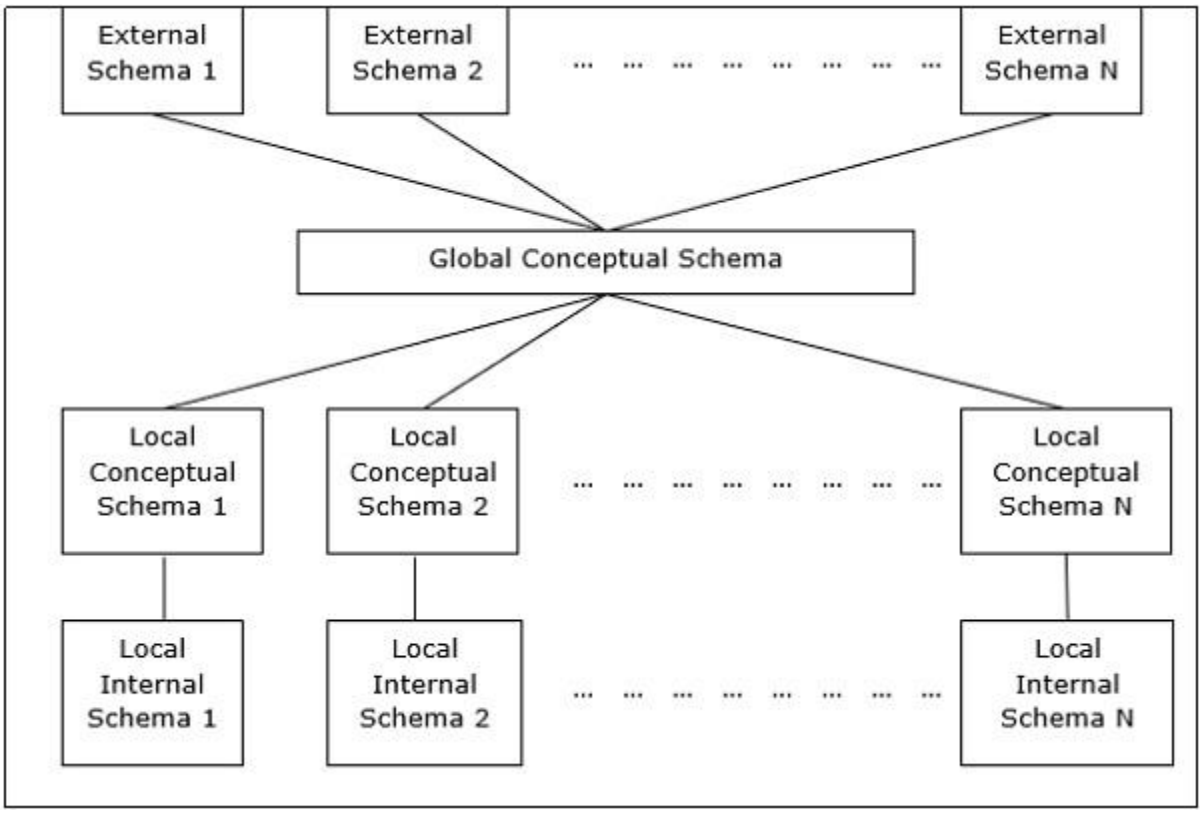


b. Peer- to-Peer Architecture for DDBMS

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 −

- **Global Conceptual Schema** − Depicts the global logical view of data.
- **Local Conceptual Schema** − Depicts logical data organization at each site.
- **Local Internal Schema** − Depicts physical data organization at each site.
- **External Schema** − Depicts user view of data.



c. Multi - DBMS Architectures

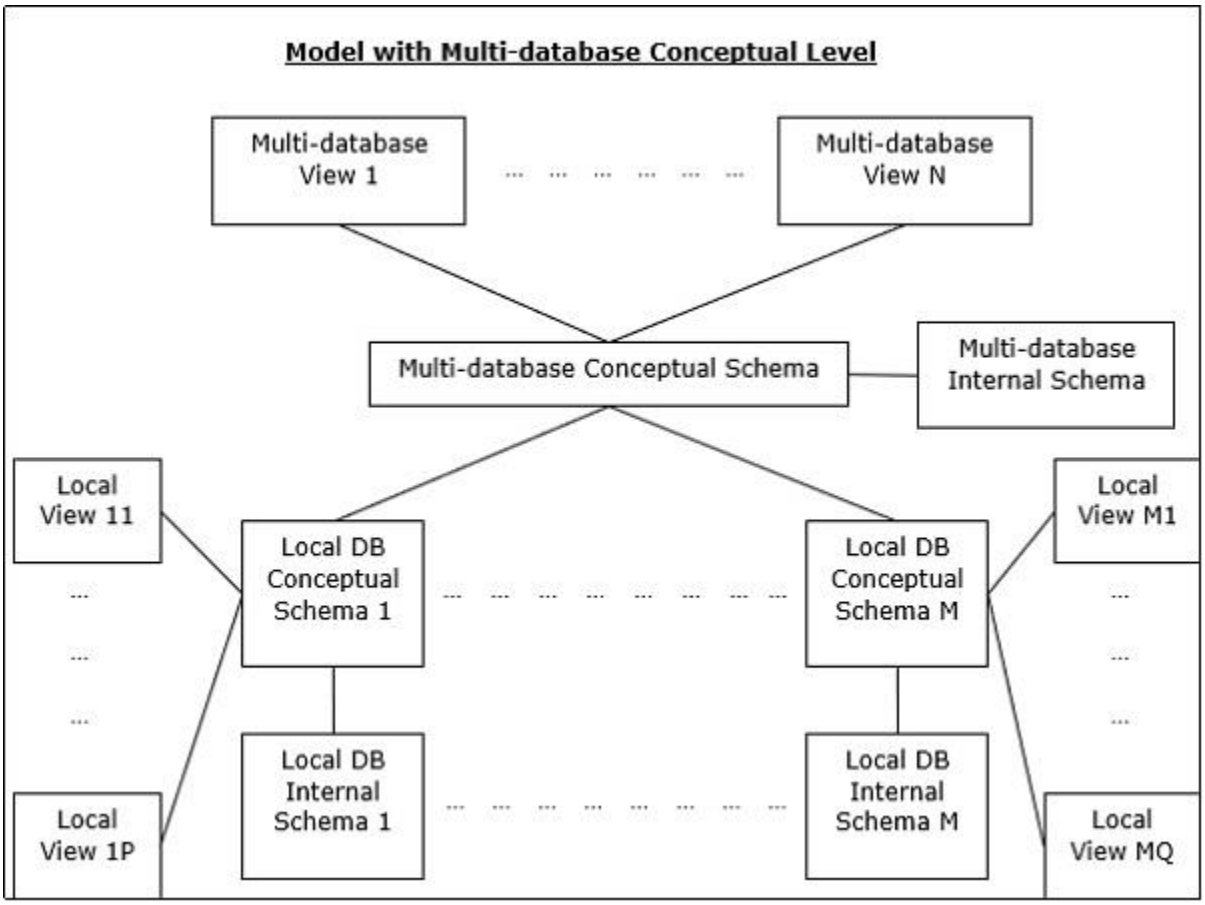
This is an integrated database system formed by a collection of two or more autonomous database systems.

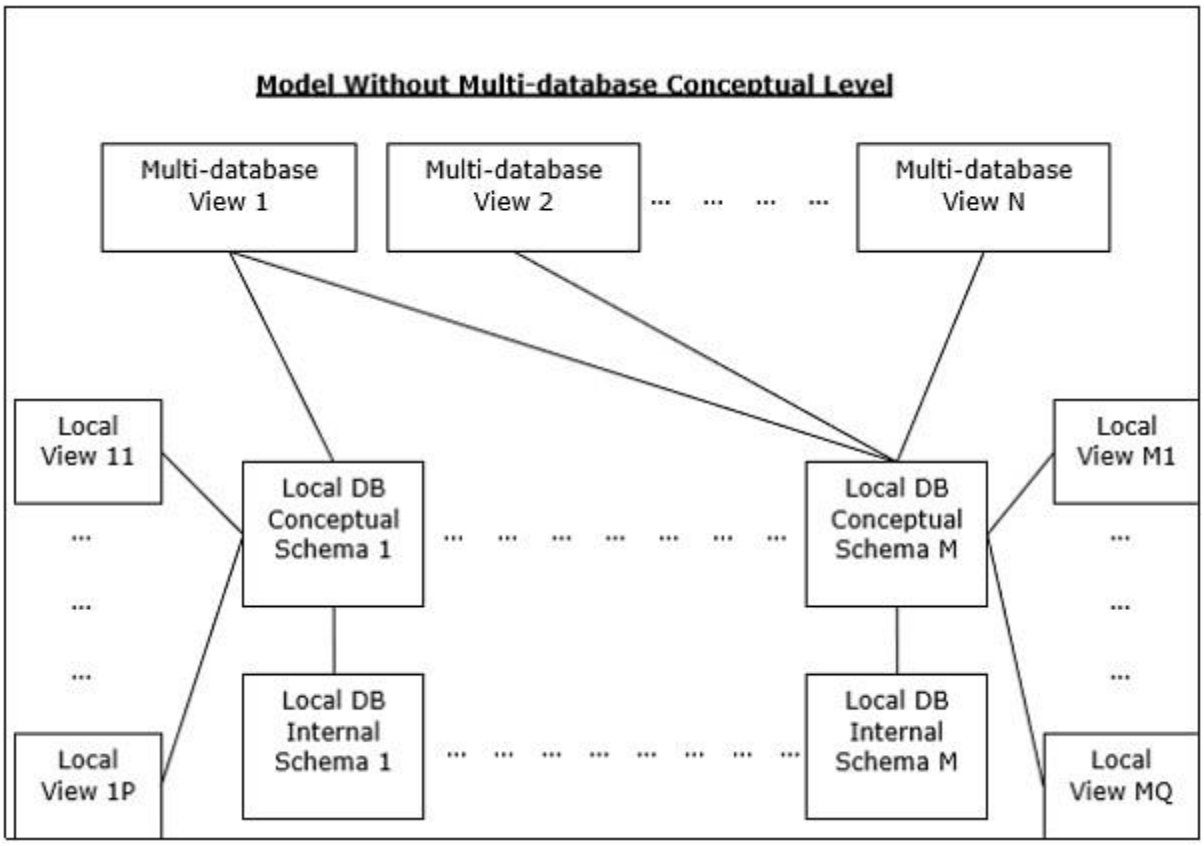
Multi-DBMS can be expressed through six levels of schemas −

- **Multi-database View Level** − Depicts multiple user views comprising of subsets of the integrated distributed database.
- **Multi-database Conceptual Level** − Depicts integrated multi-database that comprises of global logical multi-database structure definitions.
- **Multi-database Internal Level** − Depicts the data distribution across different sites and multi-database to local data mapping.
- **Local database View Level** − Depicts public view of local data.
- **Local database Conceptual Level** − Depicts local data organization at each site.
- **Local database Internal Level** − Depicts physical data organization at each site.

There are two design alternatives for multi-DBMS −

- Model with multi-database conceptual level.
- Model without multi-database conceptual level.





7.10 Design Alternatives

The distribution design alternatives for the tables in a DDBMS are as follows −

- Non-replicated and non-fragmented
- Fully replicated
- Partially replicated
- Fragmented
- Mixed

a. 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.

b. 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.

c. 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.

d. 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 −

- Vertical fragmentation
- Horizontal fragmentation
- Hybrid fragmentation

e. 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.

7.11 Distributed DBMS - Design Strategies

There are several design strategies that aid in adopting the designs. The strategies can be broadly divided into replication and fragmentation. However, in most cases, a combination of the two is used. We shall discuss each of these strategies, and its suitability.

7.11.1 Data Replication

Data replication is the process of storing separate copies of the database at two or more sites. It is a popular fault tolerance technique of distributed databases.

Advantages of Data Replication

- **Reliability** − In case of failure of any site, the database system continues to work since a copy is available at another site(s).
- **Reduction in Network Load** − Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.
- **Quicker Response** − Availability of local copies of data ensures quick query processing and consequently quick response time.
- **Simpler Transactions** − Transactions require less number of joins of tables located at different sites and minimal coordination across the network. Thus, they become simpler in nature.

Disadvantages of Data Replication

- **Increased Storage Requirements** − Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.
- **Increased Cost and Complexity of Data Updating** − Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.
- **Undesirable Application – Database coupling** − If complex update mechanisms are not used, removing data inconsistency requires complex

co-ordination at application level. This results in undesirable application – database coupling.

Some commonly used replication techniques are −

- Snapshot replication
- Near-real-time replication
- Pull replication

7.11.2 Fragmentation

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**.

Fragmentation can be of three types:

- Horizontal
- vertical, and
- hybrid (combination of horizontal and vertical).

Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called “reconstructiveness.”

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very high.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Vertical Fragmentation

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

Regd_No Name Course Address Semester Fees Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows −

```
CREATE TABLE STD_FEES AS
  SELECT Regd_No, Fees
  FROM STUDENT;
```

Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows –

```
CREATE COMP_STD AS
  SELECT * FROM STUDENT
  WHERE COURSE = "Computer Science";
```

Hybrid Fragmentation

In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways −

- At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
- At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

7.11.3 DDBMS - Distribution Transparency

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are **oblivious** of these details, they find the distributed database easy to use like any centralized database.

The three dimensions of distribution transparency are −

- Location transparency
- Fragmentation transparency
- Replication transparency

Location Transparency

Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user's site. The fact that the table or its fragments are stored at remote site in the distributed database

system, should be completely **oblivious** to the end user. The address of the remote site(s) and the access mechanisms are completely **hidden**.

In order to incorporate location transparency, DDBMS should have access to updated and accurate data dictionary and DDBMS directory which contains the details of locations of data.

Fragmentation Transparency

Fragmentation transparency enables users to query upon any table as if it were unfragmented. Thus, it **hides** the fact that the table the user is querying on is actually a fragment or union of some fragments. It also **conceals** the fact that the fragments are located at diverse sites.

This is somewhat similar to users of SQL views, where the user may not know that they are using a view of a table instead of the table itself.

Replication Transparency

Replication transparency ensures that replication of databases are **hidden** from the users. It enables users to query upon a table as if only a single copy of the table exists.

Replication transparency is associated with concurrency transparency and failure transparency. Whenever a user updates a data item, the update is reflected in all the copies of the table. However, this operation should not be known to the user. This is concurrency transparency. Also, in case of failure of a site, the user can still proceed with his queries using replicated copies without any knowledge of failure. This is failure transparency.

Combination of Transparencies

In any distributed database system, the designer should ensure that all the stated transparencies are maintained to a considerable extent. The designer may choose to fragment tables, replicate them and store them at different sites; all oblivious to the end user. However, complete distribution transparency is a tough task and requires considerable design efforts.

7.12 Date's Twelve Rules for DDBMS

The fundamental principle to the user is that a distributed system should look exactly like a non-distributed system.

Rule 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.

Rule 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

Rule 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.

Rule 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.

Rule 5: Fragmentation Independence

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

Rule 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.

Rule 7: Distributed query processing

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

Rule 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.

Rule 9: Hardware independence

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

Rule 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.

Rule 11: Network Independence

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

Rule 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

7.13 Summary

In this lecture session we defined what a distributed databases is and gave an overview of distributed databases and Distributed Database Management Systems (DDBMS). We discussed the two types of distributed database systems namely homogeneous distributed databases and heterogeneous distributed database systems. We also looked and various design issues for distributed systems including fragmentation, replication, transparency and the Dates Twelve rules for DDBMS. Next session, we shall discuss the concepts of data ware housing and data mining. Wishing you a great week ahead.

7.14 Student Activity

i. Define a distributed database system.
ii. What are the advantages of a distributed database system over a centralized database system?

iii. What are the functions of a DBMS?
iv. List and briefly discuss the components of a distributed database systems.
v. What are the design issues of distributed database environment?
vi. How do homogeneous distributed databases differ from heterogeneous distributed databases?

7.15 Reference Materials

Core Books

i. Coronel, C., & Morris, S. (2017). *Database Systems: Design, Implementation, & Management* (12th ed.). Boston, MA: Cengage Learning. ISBN: 1305627482.
ii. Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems* (7th ed.). Hoboken, NJ: Pearson Education Ltd. ISBN: 0133970779.
iii. Connolly, T. M., & Begg, C. E. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Boston, MA: Pearson Education Ltd. ISBN: 0132943263.

Core Journals

i. Journal of Database Management. ISSN: 1063-8016.
ii. Database Management & Information Retrieval. ISSN: 1862-5347.
iii. International Journal of Information Technology and Database Systems. ISSN: 2231-1807.

Recommended Text Books

i. Hernandez, M. J. (2013). *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design* (3rd ed.). Harlow, UK: Addison-Wesley. ISBN: 0321884493.
ii. Rankins, R., Bertucci, P., Gallelli, C., & Silverstein, A. (2015). *Microsoft*

SQL Server 2014 Unleashed. Indianapolis, IN: Sams Publishing. ISBN: 0672337290.

iii. Comeau, A. (2016). *MySQL Explained: Your Step By Step Guide to Database Design*. Bradenton, FL: OSTraining. ISBN: 151942437X.

Recommended Journals

i. International Journal of Intelligent Information and Database Systems. ISSN: 1751-5858.

ii. Database Systems Journal. ISSN: 2069-3230.

iii. Distributed and Parallel Databases. ISSN: 0926-8782.

iv. International Journal of Database Management Systems. ISSN: 0975 - 5985.

v. Journal of Database Management. ISSN:1063-8016.