

Department of computer science
Distributed database
Third- stage

INTRODUCTION TO DISTRIBUTED DATABASE

A database is an ordered collection of related data that is built for a specific purpose. A database may be organized as a collection of multiple tables, where a table represents a real-world element or entity. Each table has several different fields that represent the characteristic features of the entity.

For example, a company database may include tables for projects, employees, departments, products and financial records. The fields in the Employee table may be Name, Company_Id, Date_of_Joining, and so forth.

A database management system is a collection of programs that enables creation and maintenance of a database. DBMS is available as a software package that facilitates definition, construction, manipulation and sharing of data in a database. Definition of a database includes description of the structure of a database. Construction of a database involves actual storing of the data in any storage medium. Manipulation refers to the retrieving information from the database, updating the database and generating reports. Sharing of data facilitates data to be accessed by different users or programs.

Examples of DBMS Packages:

MySQL Oracle SQL Server dBASE FoxPro PostgreSQL, etc.

Database Schemas:

A database schema is a description of the database which is specified during database design and subject to infrequent alterations. It defines the organization of the data, the relationships among them, and the constraints associated with them. Databases are often represented through the three-schema architecture. The goal of this architecture is to separate the user application from the physical database. The three levels are:

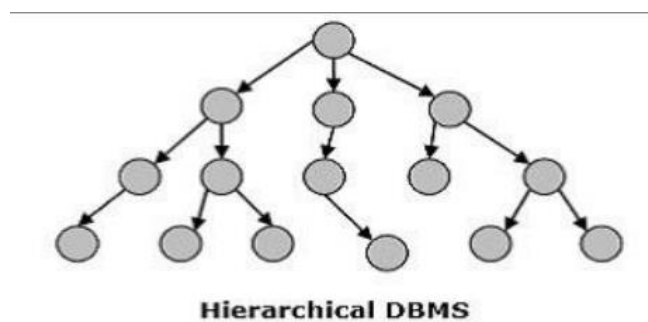
- **Internal Level** having Internal Schema: It describes the physical structure, details of internal storage and access paths for the database.
- **Conceptual Level** having Conceptual Schema – It describes the structure of the whole database while hiding the details of physical storage of data. This illustrates the entities, attributes with their data types and constraints, user operations and relationships.
- **External or View Level** having External Schemas or Views – It describes the portion of a database relevant to a particular user or a group of users while hiding the rest of database.

Department of computer science
Distributed database
Third- stage

Types of DBMS :

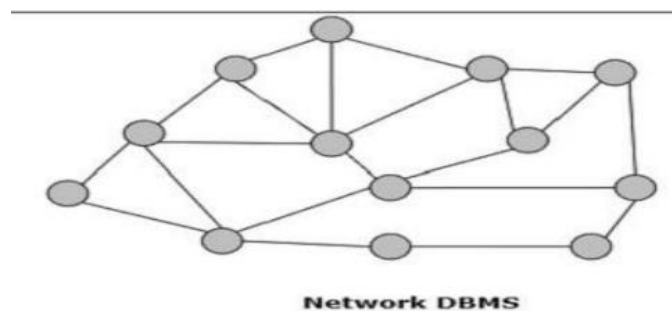
Hierarchical DBMS

In hierarchical DBMS, the relationships among data in the database are established so that one data element exists as a subordinate of another. The data elements have parent-child relationships and are modelled using the “tree” data structure. These are very fast and simple.



Network DBMS

Network DBMS is one where the relationships among data in the database are of type many-to-many in the form of a network. The structure is generally complicated due to the existence of numerous many-to-many relationships. Network DBMS is modelled using “graph” data structure.

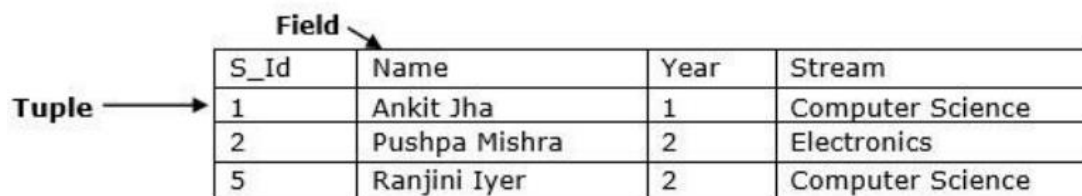


Department of computer science
Distributed database
Third- stage

Relational DBMS

In relational databases, the database is represented in the form of relations. Each relation models an entity and is represented as a table of values. In the relation or table, a row is called a tuple and denotes a single record. A column is called a field or an attribute and denotes a characteristic property of the entity. RDBMS is the most popular database management system.

For example – A Student Relation –



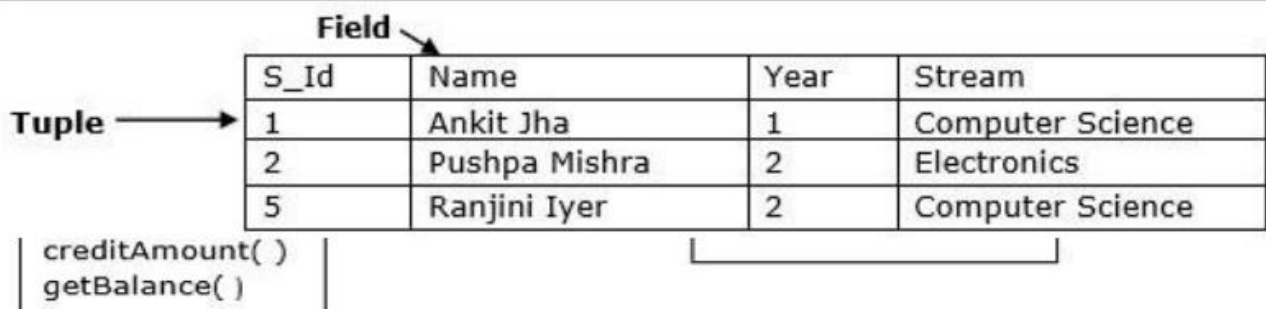
The diagram shows a table with four columns: S_Id, Name, Year, and Stream. An arrow labeled 'Field' points to the top row of the table. An arrow labeled 'Tuple' points to the first row of the table.

S_Id	Name	Year	Stream
1	Ankit Jha	1	Computer Science
2	Pushpa Mishra	2	Electronics
5	Ranjini Iyer	2	Computer Science

Object Oriented DBMS

Object-oriented DBMS is derived from the model of the object-oriented programming paradigm. They are helpful in representing both consistent data as stored in databases, as well as transient data, as found in executing programs. They use small, reusable elements called objects. Each object contains a data part and a set of operations which works upon the data. The object and its attributes are accessed through pointers instead of being stored in relational table models.

For example – A simplified Bank Account object-oriented database –



The diagram shows a table with four columns: S_Id, Name, Year, and Stream. An arrow labeled 'Field' points to the top row of the table. An arrow labeled 'Tuple' points to the first row of the table. Below the table, there is a box containing the methods: creditAmount() and getBalance().

S_Id	Name	Year	Stream
1	Ankit Jha	1	Computer Science
2	Pushpa Mishra	2	Electronics
5	Ranjini Iyer	2	Computer Science

`creditAmount()`
`getBalance()`

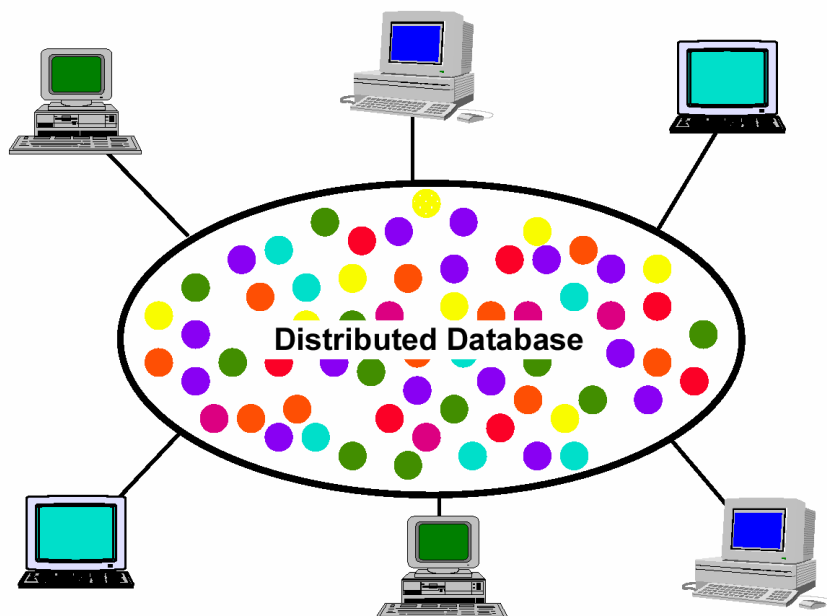
Department of computer science
Distributed database
Third- stage

Distributed DBMS

A distributed database is a set of interconnected databases that is distributed over the computer network or internet. A Distributed Database Management System (DDBMS) manages the distributed database and provides mechanisms so as to make the databases transparent to the users. In these systems, data is intentionally distributed among multiple nodes so that all computing resources of the organization can be optimally used. 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 is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.



Department of computer science
Distributed database
Third- stage

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.

Features

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.

Factors Encouraging DDBMS

- **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.
- **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.

Department of computer science
Distributed database
Third- stage

Advantages of Distributed 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.

Disadvantages of Distributed Databases

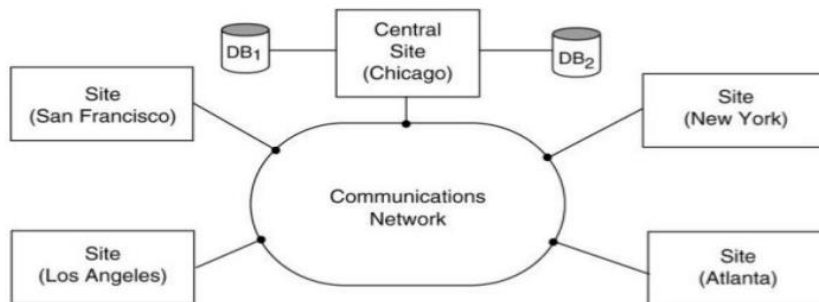
- **Need for complex and expensive software** – DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** – Even simple operations may require a large number of communications and additional calculations to provide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

Department of computer science
Distributed database
Third- stage

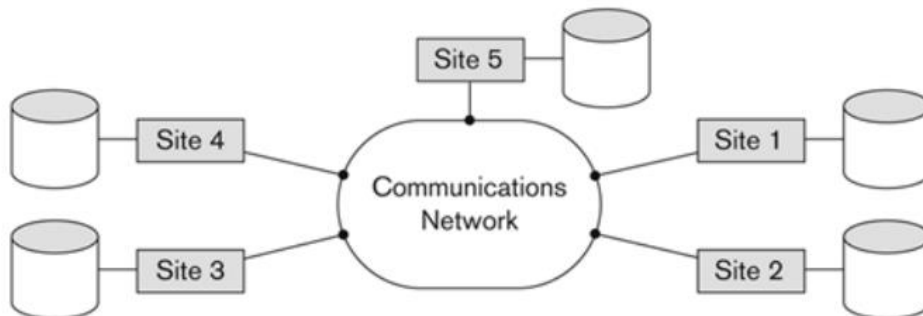
Distributed Database Vs Centralized Database

<i>Centralized DBMS</i>	<i>Distributed DBMS</i>
In Centralized DBMS the database are stored in a only one site	In Distributed DBMS the database are stored in different site and help of network it can access it
If the data is stored at a single computer site,which can be used by multiple users	Database and DBMS software distributed over many sites,connected by a computer network
Database is maintained at one site	Database is maintained at a number of different sites
If centralized system fails,entire system is halted	If one system fails,system continues work with other site
It is a less reliable	It is a more reliable

Centralized database



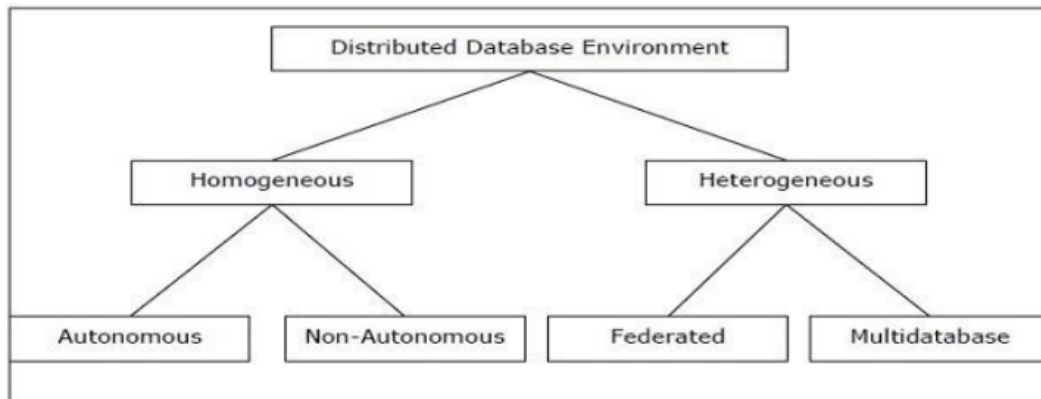
Distributed database



Department of computer science
Distributed database
Third- stage

Types of Distributed Databases

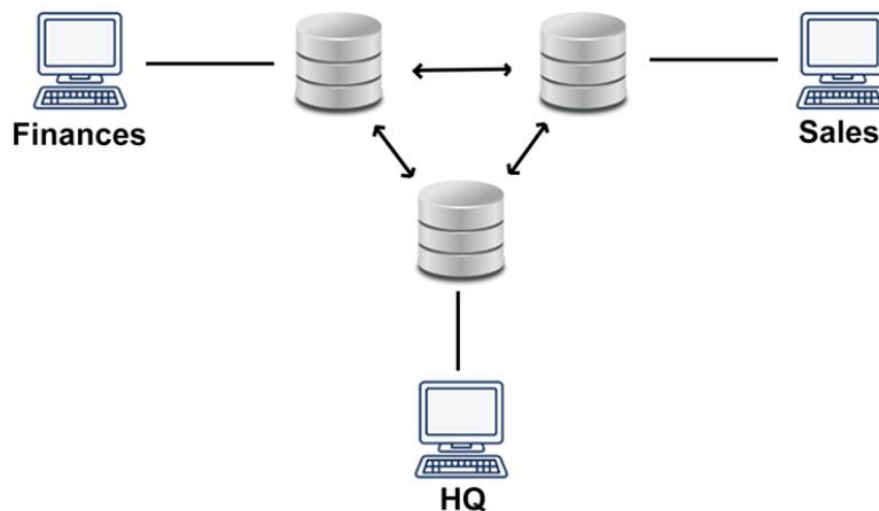
Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments



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.



Department of computer science
Distributed database
Third- stage

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database:

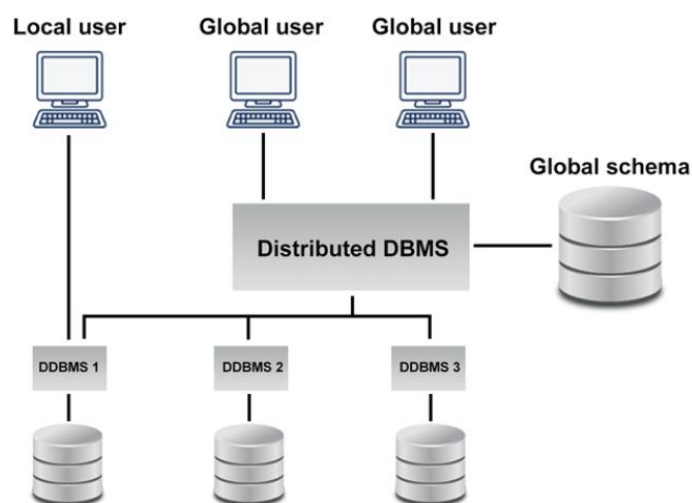
-Autonomous: Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.

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

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.



Department of computer science
Distributed database
Third- stage

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.

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.

Architectural Models

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

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. Distinguish the functionality and divide these functions into two classes, server functions and client functions.

Department of computer science
Distributed database
Third- stage

Server does most of the data management work

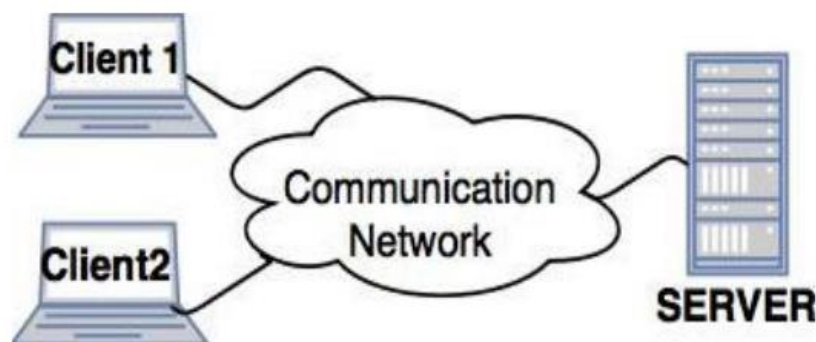
1. query processing
2. data management
3. Optimization
4. Transaction management etc

Client performs :

1. Application
2. User interface
3. DBMS Client model

The two different client - server architecture are:

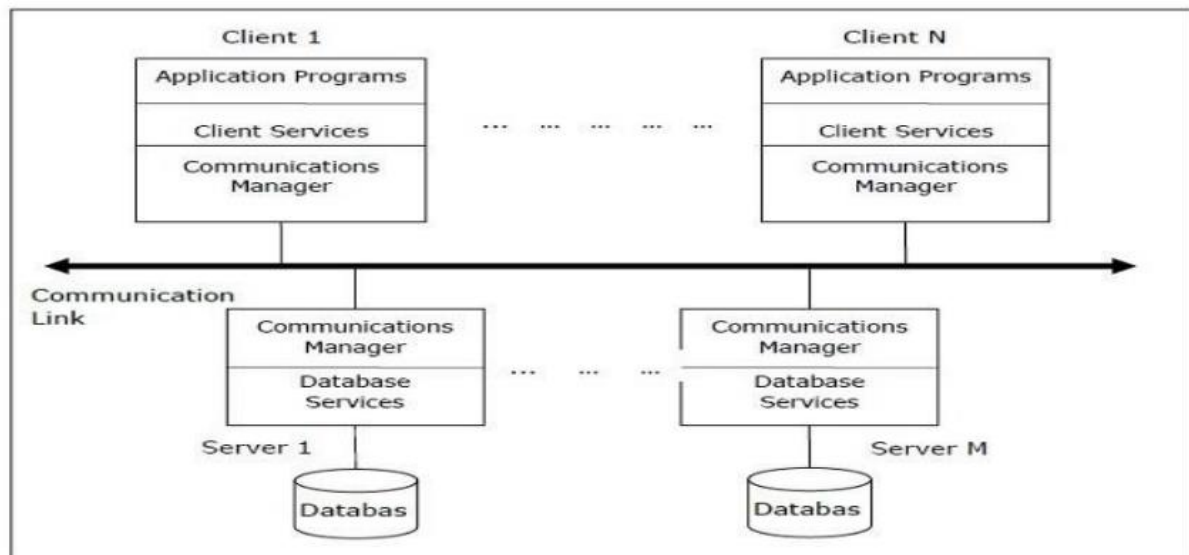
- **Single Server Multiple Client:** Single Server accessed by multiple clients
1. A client server architecture has a number of clients and a few servers connected in a network.
 2. A client sends a query to one of the servers. The earliest available server solves it and replies.
 3. A Client-server architecture is simple to implement and execute due to centralized server system.



Department of computer science
Distributed database
Third- stage

- **Multiple Server Multiple Client**

Multiple Server Multiple Client



Peer- to-Peer Architecture for DDBMS

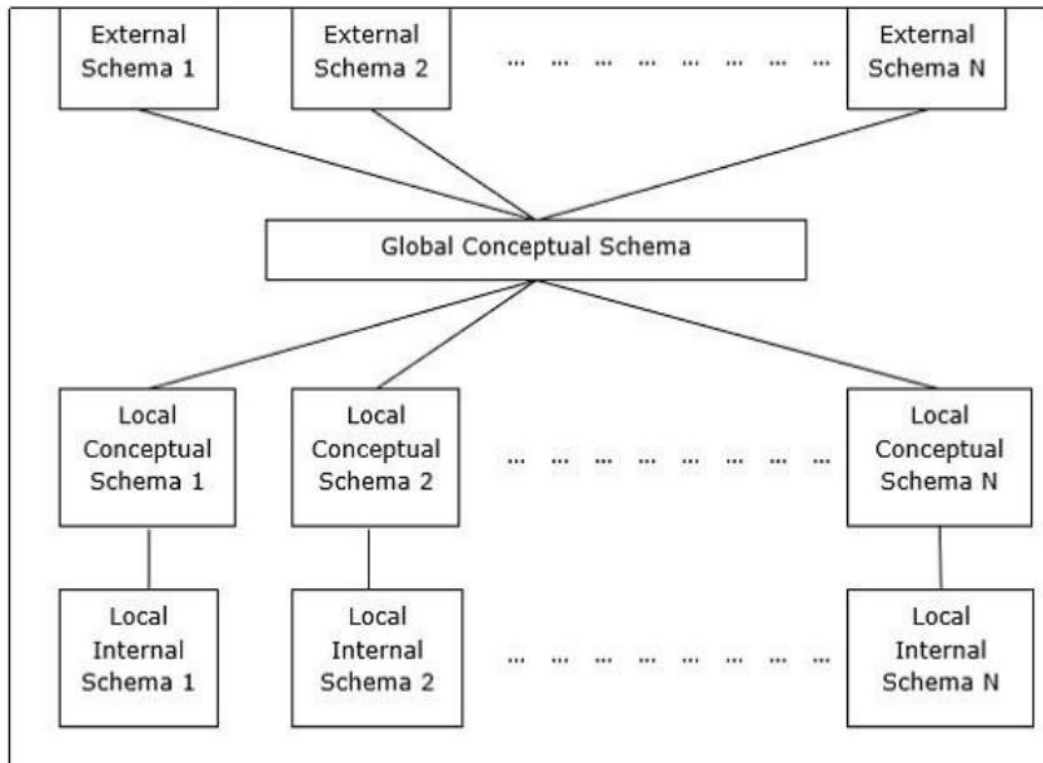
In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities. This architecture generally has four levels of schemas

- Schemas Present

- **Individual internal schema** definition at each site, **local internal schema**
- Enterprise view of data is described the **global conceptual schema**.
- Local organization of data at each site is describe in the **local conceptual schema**.
- User applications and user access to the database is supported by **external schema**.

Local conceptual schemas are mappings of the global schema onto each site. Databases are typically designed in a top-down fashion, and, therefore all external view definitions are made globally.

Department of computer science
Distributed database
Third- stage



Multi - DBMS Architectures

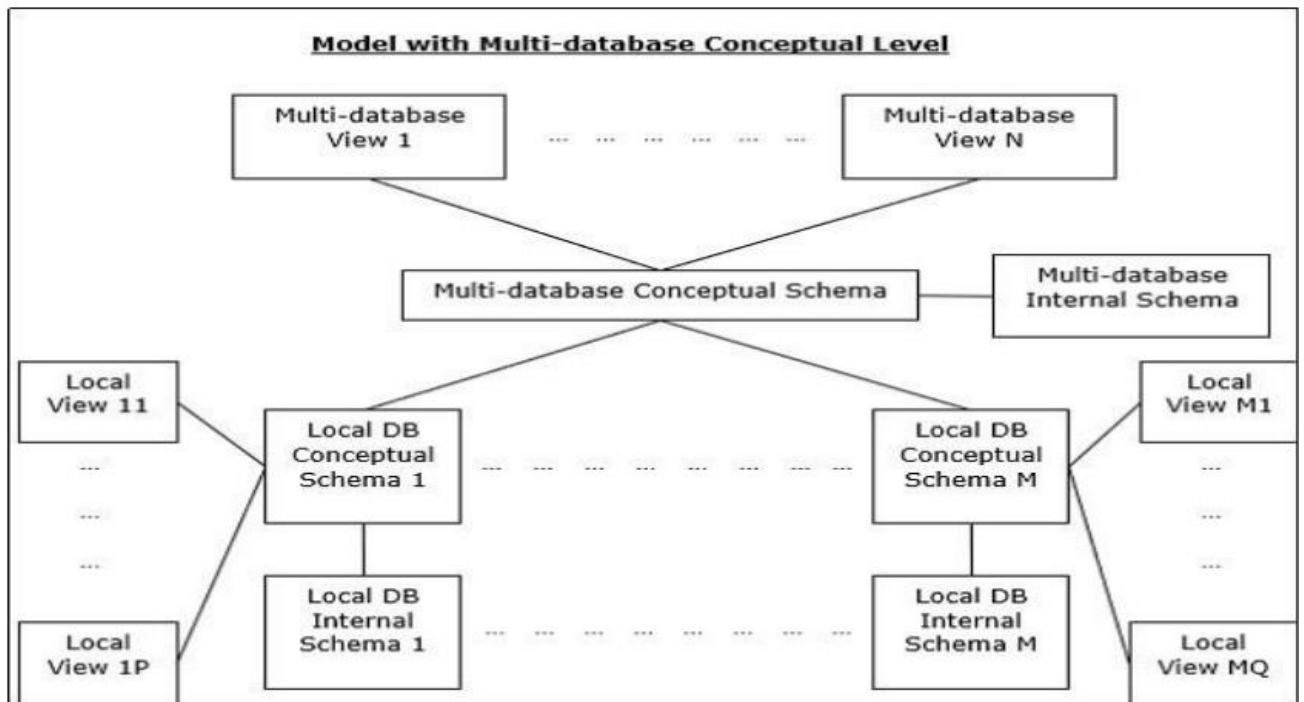
This is an integrated database system formed by a collection of two or more autonomous database systems. Multi-DBMS can be expressed through six levels of schemas – :

- 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.

Department of computer science
Distributed database
Third- stage

Models Using a Global Conceptual Schema



- GCS is defined by integrating either the external schemas of local autonomous databases or parts of their local conceptual schema
- Users of a local DBMS define their own views on the local database.
- If heterogeneity exists in the system, then two implementation alternatives exist: unilingual and multilingual
- Unilingual requires the users to utilize possibly different data models and languages
- Basic philosophy of multilingual architecture, is to permit each user to access the global database.

GCS in multi-DBMS

- Mapping is from local conceptual schema to a global schema
- Bottom-up design

Department of computer science
Distributed database
Third- stage

Model without multi-database conceptual level.

- Consists of two layers, local system layer and multi database layer.
- Local system layer , present to the multi-database layer the part of their local database they are willing share with users of other database.
- System views are constructed above this layer
- Responsibility of providing access to multiple database is delegated to the mapping between the external schemas and the local conceptual schemas.
- Full-fledged DBMs, exists each of which manages a different database.

GCS in Logically integrated distributed DBMS

- Mapping is from global schema to local conceptual schema
- Top-down procedure

Global Directory Issues

Global Directory is an extension of the normal directory, including information about the location of the fragments as well as the makeup of the fragments, for cases of distributed DBMS or a multi-DBMS, that uses a global conceptual schema,

- Relevant for distributed DBMS or a multi-DBMS that uses a global conceptual schema
- Includes information about the location of the fragments as well as the makeup of fragments.
- Directory is itself a database that contains meta-data about the actual data stored in database.

Three issues

- A directory may either be global to the entire database or local to each site.
- Directory may be maintained centrally at one site, or in a distributed fashion by distributing it over a number of sites.
 - If system is distributed, directory is always distributed
- Replication may be single copy or multiple copies.
 - Multiple copies would provide more reliability

Department of computer science
Distributed database
Third- stage

Organization of Distributed systems: Three orthogonal dimensions:

• **Level of sharing**

- No sharing, each application and data execute at one site.
- Data sharing, all the programs are replicated at other sites but not the data.
- Data-plus-program sharing, both data and program can be shared

• **Behavior of access patterns**

- Static Does not change over time Very easy to manage.
- Dynamic Most of the real life applications are dynamic

• **Level of knowledge on access pattern behavior.**

- No information
- Complete information
 - Access patterns can be reasonably predicted
 - No deviations from predictions
- Partial information
 - Deviations from predictions

Top Down Design

- Suitable for applications where database needs to be build from scratch
- Activity begins with requirement analysis
- Requirement document is input to two parallel activities:
 - view design activity, deals with defining the interfaces for end users
 - conceptual design, process by which enterprise is examined
 - Can be further divided into 2 related activity groups

Department of computer science
Distributed database
Third- stage

- Entity analyses, concerned with determining the entities, attributes and the relationship between them
- Functional analyses, concerned with determining the fun
- Distributed design activity consists of two steps:
 - Fragmentation
 - Allocation

Bottom-Up Approach

- Suitable for applications where database already exists
- Starting point is individual conceptual schemas
- Exists primarily in the context of heterogeneous database.

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

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.

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

Department of computer science
Distributed database
Third- stage

large number of queries is required to be handled whereas the number of database updates is low.

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.

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

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.

Design Strategies

In the last chapter, we had introduced different design alternatives. In this chapter, we will study the 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.

Department of computer science
Distributed database
Third- stage

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

Department of computer science
Distributed database
Third- stage

Data Storage methods for distributed databases

the best way to store data in a distributed database is to choose between **fragmentation** and **replication**.

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.”

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.

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAGCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Department of computer science
Distributed database
Third- stage

Vertical Partitions

VP1

CUSTOMER ID	FIRST NAME	LAST NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAĞCAN
4	JIM	PEPPER

VP2

CUSTOMER ID	FAVORITE COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

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.

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Department of computer science
Distributed database
Third- stage

Horizontal Partitions

HP1

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

HP2

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

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.

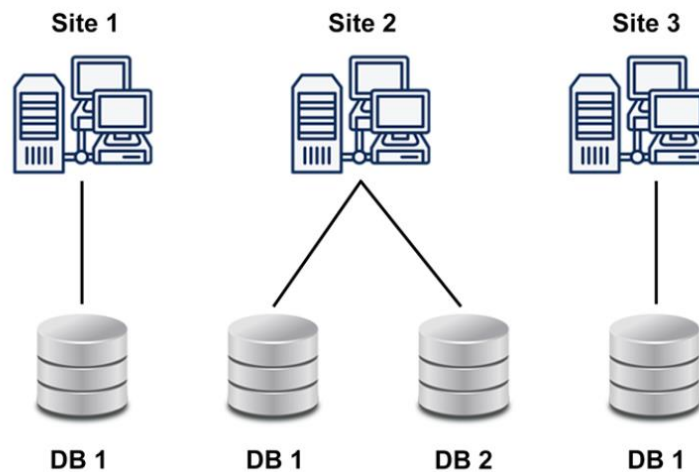
Department of computer science
Distributed database
Third- stage

Data replication

Data replication involves making copies of the data items that can reside at more than one site at any given time. Replication brings data closer to users who rely on it to make decisions and also ensures that this data is available when it is wanted.

Distributed databases make this happen by designating one of the sites as the “primary site”, and periodically syncing the other sites with the primary site. Let's say your primary site needs an upgrade, or there is an unplanned downtime event affecting your primary site — replication lets you switch users to the other sites to keep your production data available.

Your applications don't have to wait for you to spin up a new copy of an entire database, which means you won't lose transactions.



Challenges of Replication

- it requires a high degree of coordination between the different sites in a distributed database to ensure that the data values are consistent across the distributed copies.
- Additionally, with large volumes of data, more disk space is needed across the different sites, bumping up costs.

When it comes to replication speed and the consistency guarantees that replication offers, distributed databases offer two types of replication options :- *synchronous* and *asynchronous*.

Department of computer science
Distributed database
Third- stage

In the **synchronous** replication model, data written by the application to the primary site is instantly copied to all the other sites before the application is notified.

In an **asynchronous** replication model, the application gets notified as soon as data is written at the primary site, with other sites getting data lazily in the background. Eventually, all the sites are caught up with the updated data.

Another way of having your data in more than one place is by using specialized software to make copies of data and storing them offsite in case the original is lost or damaged. This is typically called duplication (or “backups”), and it is a good option for archiving old data that won't be needed too often.

Transparency

Transparency in DBMS stands for the separation of high level semantics of the system from the low-level implementation issue. High-level semantics stands for the endpoint user, and low level implementation concerns with complicated hardware implementation of data or how the data has been stored in the database. Using data independence in various layers of the database, transparency can be implemented in DBMS. 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. Unlike normal DBMS, DDBMS deals with communication network, replicas and fragments of data. Thus, transparency also involves these three factors. Following are three types of transparency:

1. Location transparency
2. Fragmentation transparency
3. Replication transparency

Department of computer science
Distributed database
Third- stage

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.

Department of computer science
Distributed database
Third- stage

Database Control

Database control refers to the task of enforcing regulations so as to provide correct data to authentic users and applications of a database. In order that correct data is available to users, all data should conform to the integrity constraints defined in the database. Besides, data should be screened away from unauthorized users so as to maintain security and privacy of the database. Database control is one of the primary tasks of the database administrator (DBA). The three dimensions of database control are :

- Authentication
- Access Control
- Integrity Constraints

AUTHORIZATION AND PROTECTION

1- Site-to-Site Protection :The first security problem which arises in a distributed database is initiating and protecting intersite communication. When two database sites communicate, it is important to make sure that:

1. At the other side of the communication line is the intended site (and not an intruder).
2. No intruder can either read or manipulate the messages which are exchanged between the sites.

The first requirement can be accomplished by establishing an identification protocol between remote sites. When two remote databases communicate with each other, on the first request they also send each other a password. When two sites decide to share some data they follow a specific mechanism.

The second requirement is to protect the content of transmitted messages once the two identified sites start to communicate. Messages in a computer network are typically routed along paths which involve several intermediate nodes and transmissions, with intermediate buffering.

The best solution to this problem consists of using cryptography, a standard technique commonly used in distributed information systems, for instance for protecting communications between terminals and processing units. Messages ("plaintext") are initially encoded into cipher messages ("ciphertext") at the sender site, then transmitted in the network, and finally decoded at the receiver site.

Department of computer science
Distributed database
Third- stage

2- User Identification :

When a user connects to the database system, they must be identified by the system. The identification is a crucial aspect of preserving security, because if an intruder could pretend to be a valid user, then security would be violated.

In a distributed database, users could identify themselves at any site of the distributed database. However, this feature can be implemented in two ways which both show negative aspects.

1. **Passwords** could be replicated at all the sites of the distributed database. This would allow user identification to be performed locally at each site, but would also compromise the security of passwords, since it would they easier for an intruder to access them.

2. **Users** could each have a "home" site where their identification is performed; in this scenario, a user connecting to a different site would be identified by sending a request to the home site and letting this site perform the identification.

A reasonable solution is to restrict each user to identifying themselves at the home site. This solution is consistent with the idea that users seem to be more "static" than, for instance, data or programs. A "pass-through" facility could be used to allow users at remote sites to connect their terminals to their "home" sites in order to identify themselves.

3- Enforcing Authorization Rules

Once users are properly identified, database systems can use authorization rules to regulate the actions performed upon database objects by them. In a distributed environment, additional problems include the allocation of these rules, which are part of the catalog, and the distribution of the mechanisms used for enforcing them. Two alternative, possible solutions are:

1. Full replication of authorization rules. This solution is consistent with having fully replicated catalogs, and requires mechanisms for distributing online updates to them. But, this solution allows authorization to be checked either at the beginning of compilation or at the beginning of execution.

Department of computer science
Distributed database
Third- stage

2. Allocation of authorization rules at the same sites as the objects to which they refer. This solution is consistent with local catalogs and does not incur the update overhead as in the first case.

The second solution is consistent with site autonomy, while the first is consistent with considering a distributed database as a single system.

The authorizations that can be given to users of a centralized database include the abilities of reading, inserting, creating, and deleting object instances (tuples) and of creating and deleting objects (relations or fragments).

4- Classes of Users

For simplifying the mechanisms which deal with authorization and the amount of stored information, individual users are grouped into classes, which are all granted the same privileges. In distributed databases, the following considerations apply to classes of users:

1. A "natural" classification of users is the one which is induced by the distribution of the database to different sites. It is likely that "all users at site x" have some common properties from the viewpoint of authorization. An explicit naming mechanism for this class should be provided.

2. Several interesting problems arise when groups of users include users from multiple sites. Problems are particularly complex when multiple-site user groups are considered in the context of site autonomy. So, mechanisms involve the consensus of the majority or of the totality of involved sites, or a decision made by a higher-level administrator. So, multiple-site user groups contrast with pure site autonomy.

Department of computer science
Distributed database
Third- stage

Query and optimization

Query: is a request for data or information from a database table or combination of tables.

Query optimization:- is the process of choosing the most appropriate execution strategy for query processing.

Global Queries to Fragment Queries

When a query is placed, it is at first scanned, parsed and validated. An internal representation of the query is then created. Then alternative execution strategies are devised for retrieving results from the database tables.

Query Optimization Issues in DDBMS

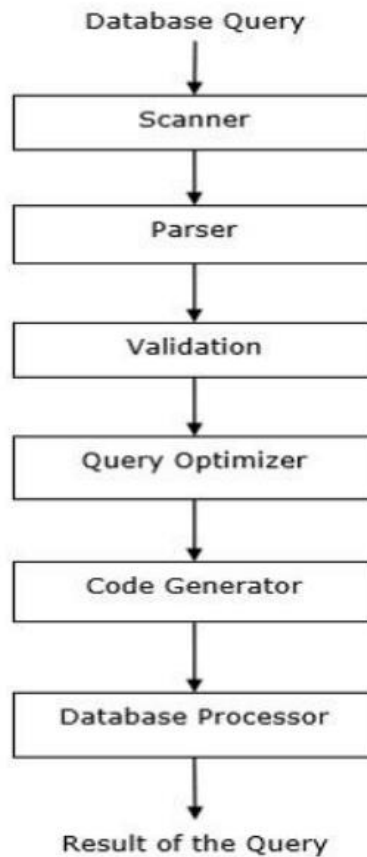
In DDBMS, query optimization is a crucial task. The complexity is high since number of alternative strategies may increase exponentially due to the following factors :

- The presence of a number of fragments.
 - Distribution of the fragments or tables across various sites.
 - The speed of communication links.
 - Disparity in local processing capabilities. Hence, in a distributed system, the target is often to find a good execution strategy for query processing rather than the best one. The time to execute a query is the sum of the following
- Time to communicate queries to databases.
 - Time to execute local query fragments.
 - Time to assemble data from different sites.
 - Time to display results to the application.

Department of computer science
Distributed database
Third- stage

Query Processing

Query processing is a set of all activities starting from query placement to displaying the results of the query. The steps are as shown in the following diagram –



MANAGEMENT OF DISTRIBUTED TRANSACTIONS

- it is a framework for Transaction

A transaction is a program including a collection of database operations, executed as a logical unit of data processing. The operations performed in a transaction include one or more of database operations like insert, delete, update or retrieve data.

Department of computer science
Distributed database
Third- stage

Transaction Operations

The low level operations performed in a transaction are –

- begin_transaction – A marker that specifies start of transaction execution.
- read_item or write_item – Database operations that may be interleaved with main memory operations as a part of transaction.
- end_transaction – A marker that specifies end of transaction.
- commit – A signal to specify that the transaction has been successfully completed in its entirety and will not be undone.
- rollback – A signal to specify that the transaction has been unsuccessful and so all temporary changes in the database are undone. A committed transaction cannot be rolled back.

Desirable Properties of Transactions

Any transaction must maintain the ACID properties, viz. Atomicity, Consistency, Isolation, and Durability.

- Atomicity – This property states that a transaction is an atomic unit of processing, that is, either it is performed in its entirety or not performed at all. No partial update should exist.
- Consistency – A transaction should take the database from one consistent state to another consistent state. It should not adversely affect any data item in the database.
- Isolation – A transaction should be executed as if it is the only one in the system. There should not be any interference from the other concurrent transactions that are simultaneously running.
- Durability – If a committed transaction brings about a change, that change should be durable in the database and not lost in case of any failure.

Department of computer science
Distributed database
Third- stage

States of a transaction

Active: Initial state and during the execution

Partially committed: After the final statement has been executed

Committed: After successful completion

Failed: After the discovery that normal execution can no longer proceed

Aborted: After the transaction has been rolled back and the DB restored to its state prior to the start of the transaction. Restart it again or kill it.

Goal: The goal of transaction management in a distributed database is to control the execution of transactions so that:

Transactions have atomicity, durability, serializability and isolation properties. • CPU and main memory utilization • Control messages • Response time • Availability

Distributed Transactions A distributed transaction is a database transaction in which two or more network hosts are involved. Usually, hosts provide transactional resources, while the transaction manager is responsible for creating and managing a global transaction that encompasses all operations against such resources.

Transaction Control

Transaction control is concerned with designating and controlling the sites required for processing a transaction in a distributed database system. There are many options regarding the choice of where to process the transaction and how to designate the center of control, like :

- One server may be selected as the center of control.
- The center of control may travel from one server to another.
- The responsibility of controlling may be shared by a number of servers.

Department of computer science
Distributed database
Third- stage

Distributed Deadlock Avoidance

As in centralized system, distributed deadlock avoidance handles deadlock prior to occurrence. Additionally, in distributed systems, transaction location and transaction control issues needs to be addressed. Due to the distributed nature of the transaction, the following conflicts may occur :

- Conflict between two transactions in the same site.
- Conflict between two transactions in different sites.