Distributed and Client Server Databases

Unit 4

Red : Indicates important

Contents

- Distributed database system
- Data Replication and fragmentation
- Client Server databases

Need For Distributed Database Systems

- A distributed database is a set of database stored on multiple computers. It appears to the application as a single database.
- An application can access and modify the data in several databases within a network simultaneously.
- Each database is controlled by its local server and maintains the consistency of the global distributed database.
- The computers in a distributed system communicates with each other through various communication media like high-speed buses or telephone line.
- It doesn't share main memory nor a clock. To work properly many computers have to synchronise their clocks.
- In some cases, absolute time is more important and the clocks are synchronised with atomic clocks.

- The processors in a distributed system vary in size and function. It may be small microcomputers, workstations, minicomputers, and large general-purpose computer system.
- The processors are referred using terms like sites, nodes, computers and so on depending on the context in which they are mentioned. We mainly use the term site.
- A distributed database system consists of a collection of sites. Each site participates in the execution of transactions, which access data at one site, or several sites.
- Difference between centralised and distributed database systems is, in centralised system data resides in one single centralised control whereas in distributed database system the data resides in several sites under the control of local distributed DBMS components which are under the control of one DBMS.

- In a centralised database the DBMS and data reside at a single database instance.
- For recovery purpose, redundant database information which is under the control of a single DBMS.
- An enhancement of centralised database system is to provide access to centralised data from a number of distributed locations through a network. In this system, a site failure except the central site will not lead to total system failure.
- Some of the problems with centralised database :
  - Loss of messages between a local and central site.
  - Failure of communication link between local and central site. This will make the system unreliable.
  - Excessive load on the central site will delay queries and accesses. A single site would have to bear a large number of transactions and this would require large computing systems.

- The above mentioned problems can be overcome by using distributed database systems.
- Distributed database systems improve the reliability and sharability of data and the efficiency of data access.
- Distributed database systems can be considered as a system connected to intelligent remote devices each of which acts as a local database repository. All data is accessible from each site. It increases the efficiency of access because multiple sites can co-ordinate efficiently to respond to a query and control and processing is limited to this DBMS.

Structure Of Distributed Database

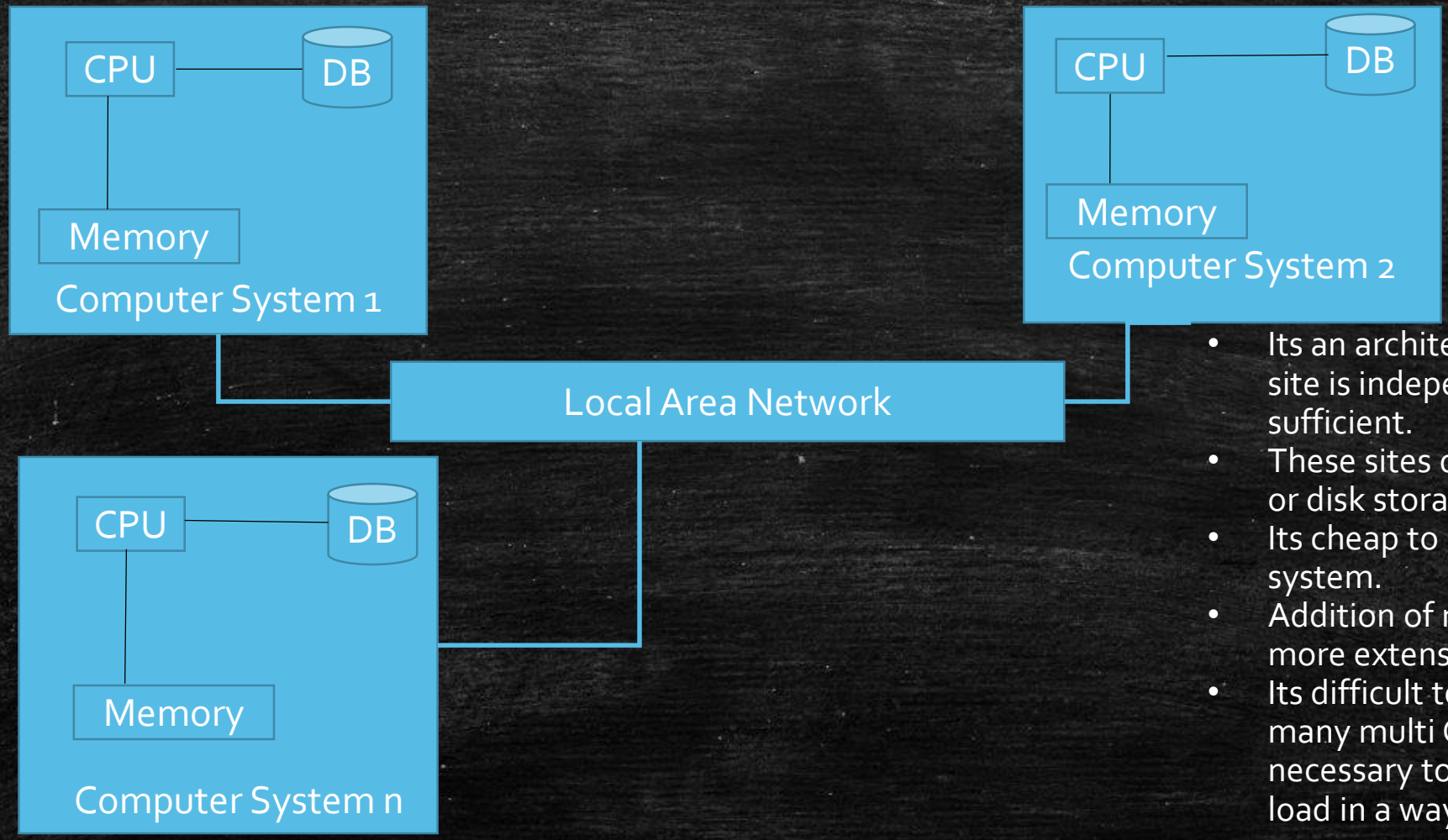
- Distributed database system consists of a collection of sites, each of which maintains a local database system.
- Each system will be able process local transactions(transactions that access data only in that single site).
- It may also participate in execution of global transactions(transactions that access data at several sites).
- Three different database system architectures is shown in figure 1.

1a) No database sharing architecture

1b) A networked architecture with a centralised database.

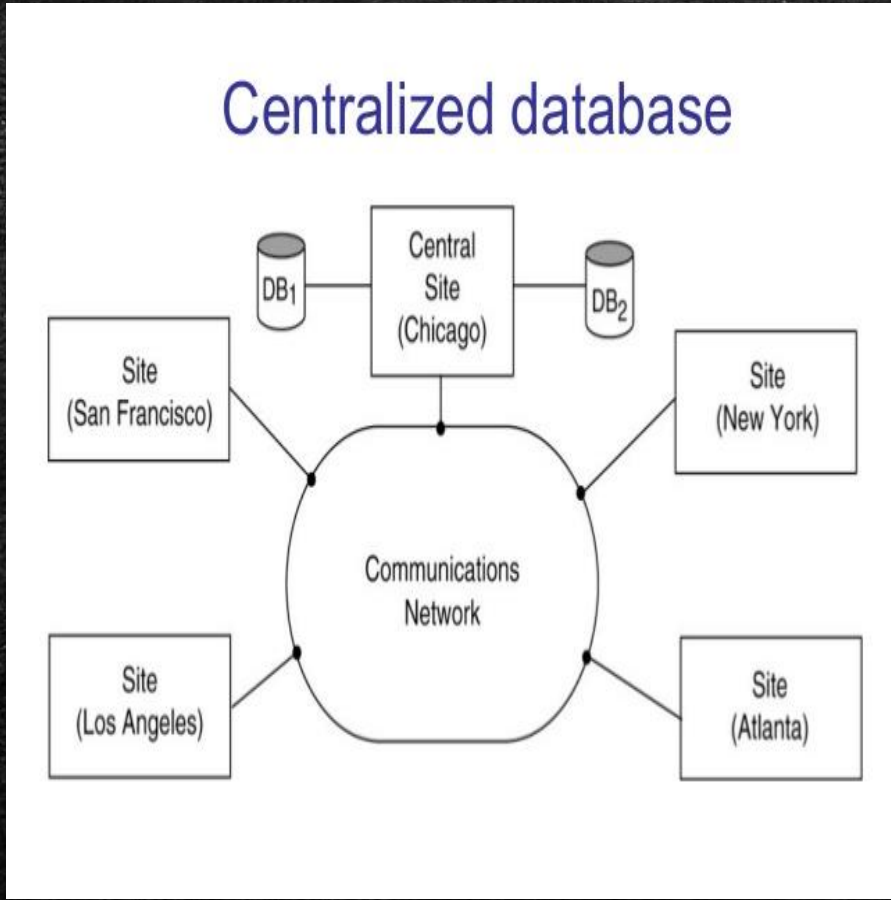
1c) A distributed database architecture

No Database Sharing Architecture (1a)



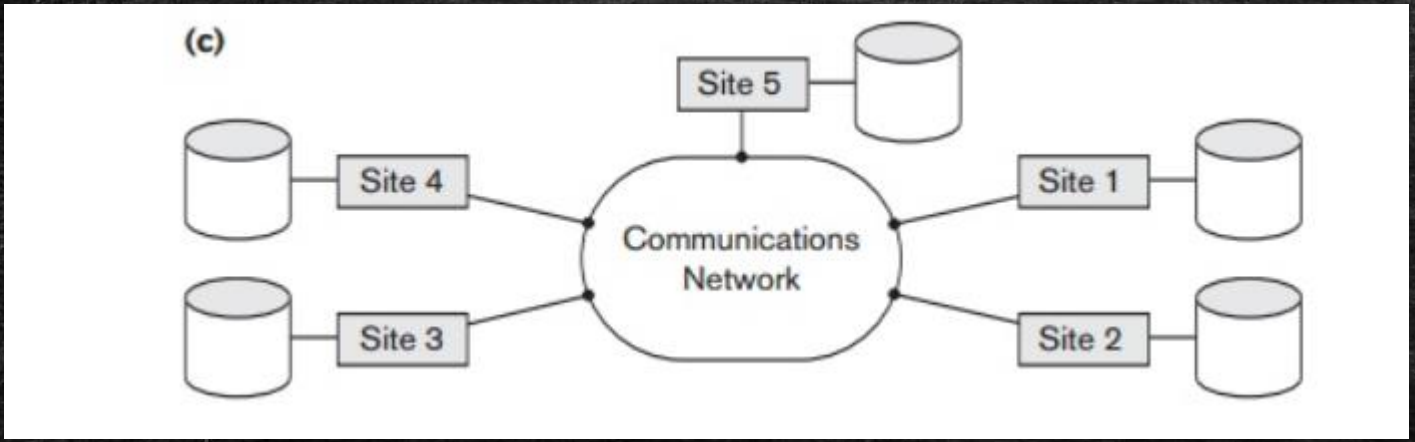
- Its an architecture in which each site is independent and self-sufficient.
- These sites do not share memory or disk storage.
- Its cheap to build this type of system.
- Addition of new nodes requires more extensive reorganisation.
- Its difficult to load-balance. In many multi CPU environments, its necessary to split the system work load in a way so that all system resources are used efficiently. It is difficult to achieve this.

A Networked Architecture With A Centralised Database



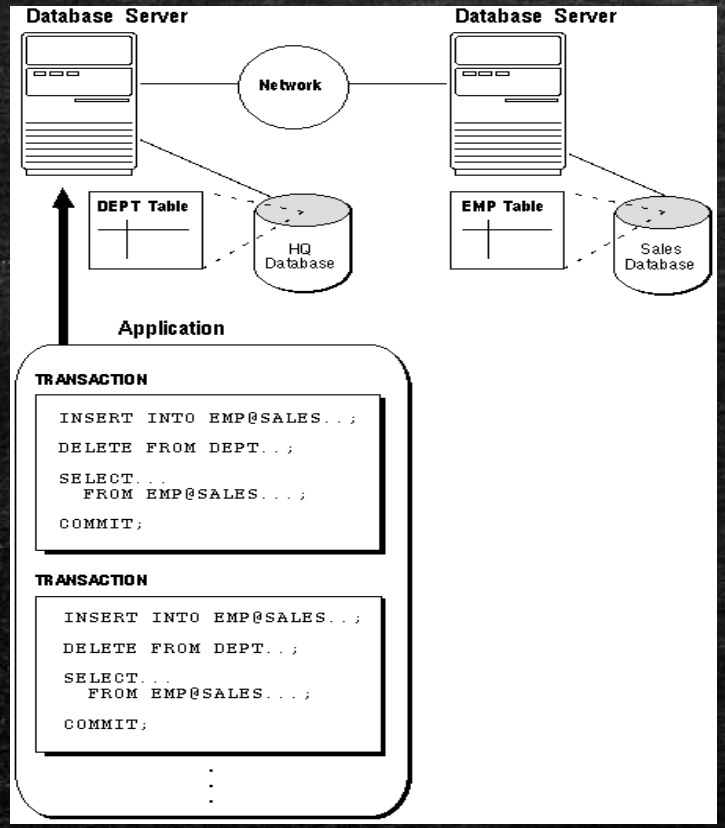
- A **centralized database** is a database that is located, stored, and maintained in a single location. This is usually a mainframe computer or a server.
- It's a traditional approach for storing data in large organizations or institutions.
- It does not interact with other computer systems.
- Memory or storage disk is shared.
- User access systems via computer terminals that had no processing power and had only display capabilities.
- The centralised database should be able to satisfy all the requests coming to the system, hence this may lead to bottleneck.
- All the data resides in one location so it is easy to maintain and backup data.
- Its easy to maintain data integrity, once data is stored in central site, outdated data is no longer available in other sites.

A Distributed Database Architecture (1c)



- For notes on this write need for distributed database system discussed previously.
- The execution of global transactions requires communications among the sites.
- Figure 2 in the next slide illustrates distributed database system architecture having transactions.

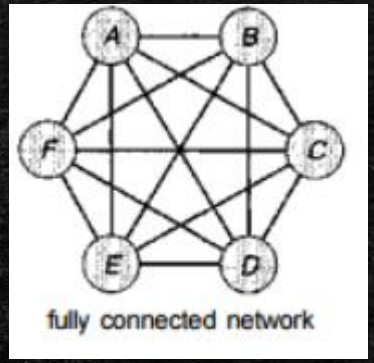
Distributed Database Schema And Architecture



Issues involved in DDBMS are :

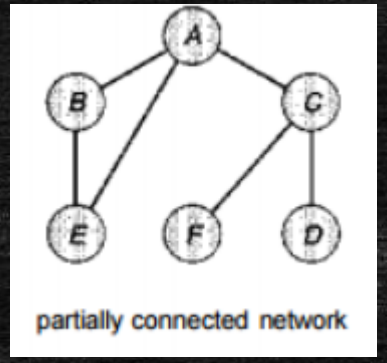
- How the data is distributed?
- Is this data distribution hidden from general users?
- How is data integrity and security maintained locally and globally?
- How are the distributed database connected?

- The sites in a distributed database system can be connected physically in different ways.
- The various topologies are represented as graphs where nodes represent the sites. An edge from node A to node B represents a direct connection between the two sites.
- Some of the most common structures used are shown in figures below :



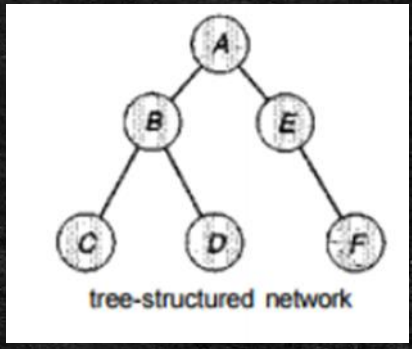
fully connected network

- This structure is very reliable but expensive.



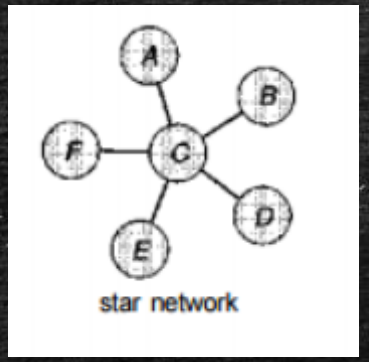
partially connected network

- This structure may lead to network partitions.



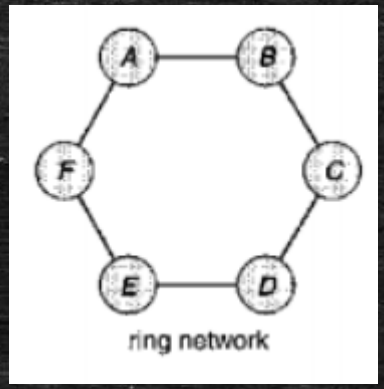
tree-structured network

- This structure may require heavy load on the root i.e. node A.



star network

This structure is totally depend on the central link i.e. if the central computer or hub fails, the entire network goes down and all computers are disconnected from the network.



ring network

- In this structure we need to check for ring failure due to one site i.e. one malfunctioning site can create problems for the entire network.

- Differences among these structures is based on the following factors :
- Installation cost : The cost of physically linking the sites in the system
- Communication cost : The cost in terms of time and money to send a message from one site to another site.
- Reliability : The frequency with which a link or site fails.
- Availability : The degree to which data can be accessed even though there is failure of links or sites.
- The sites of a network may be distributed physically either over a large geographical area (like all over India) or over a small geographical area (like within a building or a number of adjacent buildings).
- Large geographical are based on wide area network and small geographical area are based on local area network.

- The communication links in a wide area networks are relatively slow and less reliable compared to local area networks. The most common communication links used in WAN are telephone lines, microwave links and satellite channels.
- The communication links in a local area network are of higher speed and lower error rate compared to wide area networks. The most common links used are twisted pair, base band coaxial and fibre optics.

A Distributed Transaction

- Now we will discuss distributed transaction by considering a banking system that consists of three branches located in three different cities. Each branch has its own computer with a database consisting of all the accounts maintained at that branch.
- There exists a single site which stores information about all the other branches of the bank.
- Assume that the database systems at various sites are based on relational model.
- Each branch maintains its portion of relation : DEPOSIT (DEPOSIT- BRANCH) where

  DEPOSIT-BRANCH = (branch-name, account number, customer-name, balance)
- A single site contains information about all the other branches of the bank.

  BRANCH-DETAILS(branch-name, Financial_health, city)

- A local transaction is a type of transaction that accesses accounts in one single site, at which the transaction was initiated. E.g. transaction to add Rs. 5000 to account number 123 located at Chennai branch.
- A global transaction is a type of transaction that accesses accounts in a site different from the one at which it was initiated, or accesses accounts in several different sites. E.g. Transaction to transfer Rs. 5000 from account 123 to account 410 , which is located at Delhi branch.

Advantages And Disadvantages

- There are several advantages of building distributed database systems.

1. Data Sharing and Distributed Control

There is a provision in the environment where user at one site may be able to access the data residing at other sites. The main advantage of this is that the user need not know the site from which data is being accessed. Data is placed at the site close to the users who normally use the data. A global database administrator (DBA) is responsible for the whole system. Part of this responsibility is given to the local administrator (DBA) can manage the local DBMS.

The primary advantage of data sharing by means of data distribution each site is able to retain a degree of control over data stored locally.

2. Reflects organisational structure

Many organizations are distributed over several locations. For example, a bank has many offices in different cities. In such cases database will also be distributed.

Advantages Of DDBMS

A bank may keep a database at each branch office containing details such things as the staff that work at that location, the account information of customers etc.

The staff at a branch office will make local inquiries of the database. The company headquarters may wish to make global inquiries involving the access of data at all or a number of branches.

3. Improved Reliability

In a centralised DBMS, a server failure terminates the operations of the DBMS. In DDBMS, a failure at one site or failure of communication link makes some sites inaccessible and does not make the entire system inoperable.

DDBMS are designed in such a way that it continues to function even if there is a failure. If data are replicated in different sites, a transaction requiring a particular data may find at different sites. Thus, failure does not imply shutdown of the system.

The recovery from failure is more complex in DDBMS than in centralised system.

The failure of one site must be detected by the system and appropriate action may be needed to recover from the failure. The system must no longer use the services of the failed site. Finally, when the failed site is repaired, mechanisms must be available to integrate it smoothly back to the system.

4. Improved Availability

In distributed database system, data is replicated in several sites. If one site fails in a distributed system the remaining sites may be able to continue operating. This increases the availability which is crucial for database systems in real life applications. If data are replicated in different sites, a transaction requiring a particular data may find at different sites. Thus, failure does not imply shutdown of the system.

5. Improved Performance

A DDBMS fragments the database to keep data closer to where it is most needed. This reduces the management (data access and modification) time significantly. Each site handles only a part of the entire database, there may not be the same contention for CPU and I/O services as in centralised DBMS.

6. Speedup Query Processing

A query that involves data at several sites is split into sub-queries. These sub-queries are executed in parallel by several sites. This allows faster processing of a query. In cases where the data is replicated, queries are sent to the least heavily loaded sites.

7. Economics

It is now generally accepted that it costs much less to create a system of smaller computers with the equivalent power of a single large computer. This makes it more cost effective for corporate divisions and departments to obtain separate computers. It is also much more cost-effective to add workstations· to a network than to update a mainframe system.

The second potential cost saving occurs where database are geographically remote and the applications require access to distributed data. In such cases, owing to the relative expense of data being transmitted across the network as opposed to the cost of local access, it may be much more economical to partition the application and perform the processing locally at each site.

8. Modular Growth

In a distributed environment, it is much easier to handle expansion. New sites can be added to the network without affecting the operations of other sites. This flexibility allows an organization to expand relatively easily. Adding processing and storage power to the network can usually handle the increase in database size. In a centralized DBMS, growth may require changes to both hardware and software. Removal of a site also does not cause much problem.

Disadvantages Of DDBMS

- The primary disadvantage is the added complexity needed to ensure proper coordination among sites. Some of the disadvantages are :

1. **Cost**

Distributed database systems are complex to implement and thus more costly. Increased complexity means that we can expect the procurement and maintenance costs *for* a DDBMS to be higher than those for a centralized DBMS.

Distributed DBMS requires additional hardware to establish a network between sites. There are ongoing communication costs incurred with the use of this network. There are also additional maintenance costs to manage and maintain the local DBMSs and the underlying network.

2. Greater potential for bugs

The sites of a distributed system operate concurrently so, its more difficult to ensure the correctness of algorithm. The art of constructing distributed algorithms is an active and important area of research.

3. Increased processing overhead

The exchange of messages and the additional computation needed to achieve coordination among the sites is an overhead that does not arise in centralised system.

4. Complexity

A DDBMS that is reliable, available and secure is inherently more complex than a centralised DBMS. Data replication also adds to complexity of the DDBMS. Data replication is necessary to have availability, reliability and performance.

5. Security

In a centralized system, access to the data can be easily controlled. However, in a distributed DBMS not only does access to replicated data have to be controlled in multiple locations but also the network itself has to be made secure. In the past, networks were regarded as an insecure communication medium. Although this is still partially true, significant developments have been made to make networks more secure.

6. Lack of standards and experience:

The lack of standards has significantly limited the potential of DDBMS. Also, there are no tools or methodologies to help users convert a centralised DBMS into a DDBMS.

7. Integrity control more difficult

Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. In a distributed DBMS, the communication and processing costs that are required to enforce integrity constraints are high as compared to centralized system.

5. Purpose

General-purpose DDBMS have not been widely accepted. We do not have the same level of experience in industry as we have with centralised DBMS.

6. Database design more complex

Besides the normal difficulties of designing a centralized database, the design of a distributed database has to take account of fragmentation of data, allocation of fragmentation to specific sites, and data replication.

Design of Distributed Databases

- The distributed databases are primarily relational at local level.
- The local database schema is same as that of a centralised database design.
- The strategies that aid in adopting design can be broadly divided into replication and fragmentation.

Replication :

- Its defined as a copy of a relation.
- Each replica is stored at a different site.
- It is a popular fault tolerance technique of distributed databases.
- Alternative to replication is to store only one copy of a relation and this is not recommended in distributed databases.

- If a relation R has its copies stored at two or more sites , then is considered to be replicated.
- Advantages and disadvantages of replication of relation :
  - Availability : A site containing the copy of a replicated relation fails, even then the relation may be found in another site. Thus, the system may continue to process at least the queries containing just read operation despite the failure of one site. Write operation can also be performed but with suitable recovery algorithm.
  - Increased parallelism : Since the replicated data has many copies a query can be answered from the least loaded site or can be distributed. With more replicas you have greater chances that the needed data is found on the site where the transaction is executing. Hence, data replication can minimise movement of data between sites.
  - Increased overheads on update : On the disadvantage side, it will require the system to ensure that all replicas of a relation are consistent. This means that all copies of the relation need to be updated at the same time, resulting in increased overheads. For example, in a banking system, if account information is replicated at various sites, it is necessary that the balance in a particular account should be the same at all sites.

- The problem of controlling concurrent updates by several transactions to replicated data is more complex than the centralised approach of concurrency control. The management of copies of relation can be simplified by choosing one of the copies as the primary copy. For example, in a banking system, the primary copy of an account may be the site at which the account has been opened.
- The replication can be classified as :
  - Complete replication : It means maintaining a complete copy of the database at each site. Hence, the reliability and availability and performance for query response are maximized. Storage costs, communication costs and updates are expensive. To overcome some of these problems relation, snapshots are used. A snapshot is defined as the copy of the data at a given time. Snapshots are updated periodically, such as hourly or weekly. Thus, snapshots may not be update.
  - Selective replication : It's a combination of creating small fragments of relation and replicating them rather than a whole relation. The data should be fragmented on need basis of various sites, as per the frequency of use, otherwise data is kept at a centralised site. This is the most commonly used strategy as it provides flexibility.

Fragmentation :

- Fragmentation involves decomposing the data in relation to non-overlapping component relations.
- The reasons for fragmenting a relation are :
  - Use of partial data by applications : Applications work with views rather than entire relations. Therefore it may be more appropriate to work with subsets of relations rather than entire data.
  - Increases efficiency : Data is stored close to most frequently used site, thus retrieval would be faster. Also, data that is not needed by local applications is not stored, thus the size of data to be looked into is smaller.
  - Parallelism of transaction execution : A transaction can be divided into several sub-queries that can operate on fragments in parallel. This increases the degree of concurrency in the system, thus allowing transactions to execute efficiently.
  - Security : Data not required by local applications is not stored at the site, thus so unnecessary security violations may exist.

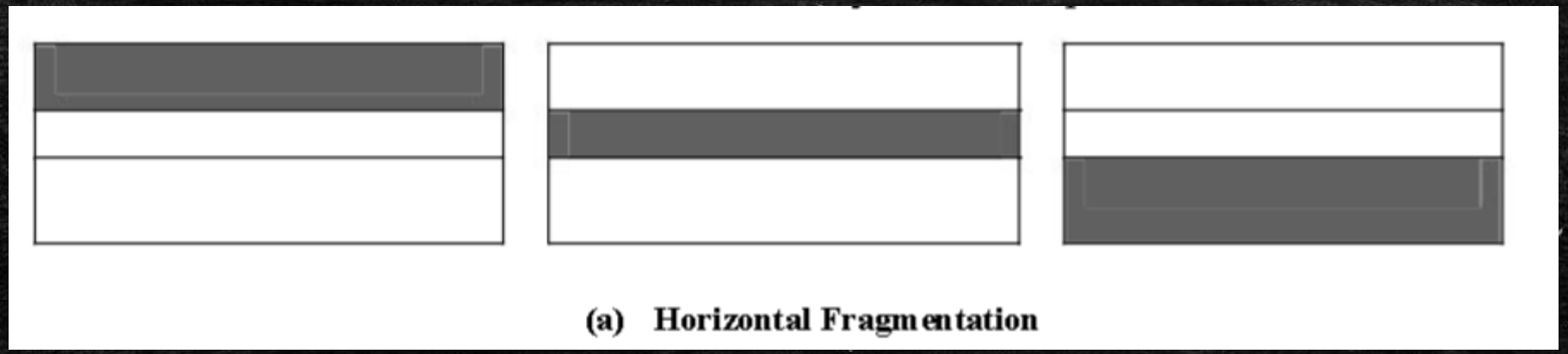
- Rules for fragmentation :

a) Completeness : This rule ensures that there is no loss of data during fragmentation. If a relation is decomposed into fragments, then each data item must appear in at least one fragment.

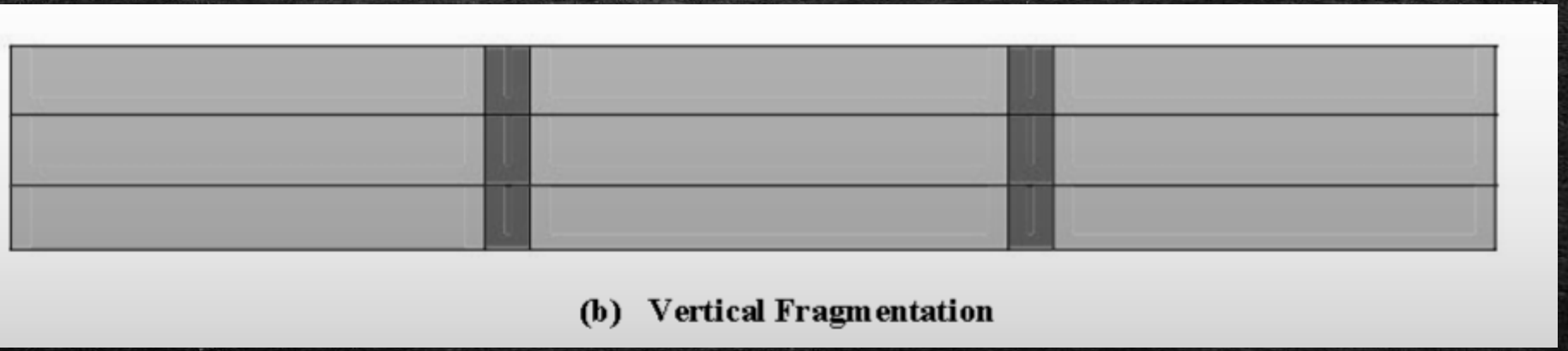
b) Reconstruction : This rule ensures preservation of functional dependencies. It should be possible to define a relational operation that will reconstruct the relation from its fragments.

c) Disjointness : A data item that appears in a fragment should not appear in any other fragment. However, vertical fragmentation is an exception to this rule. In vertical fragmentation the primary key attributes must be repeated to allow reconstruction of original relation. This rule ensures minimization of data redundancy.

- Types of fragmentation :
- There are two major types of fragmentation : horizontal and vertical fragmentation.
- Horizontal fragments are subsets of tuples and vertical fragments are subsets of attributes.
- There are other types of fragmentation : mixed, and derived – a type of horizontal fragmentation.



(a) Horizontal Fragmentation



(b) Vertical Fragmentation

Horizontal fragmentation :

- Horizontal fragmentation groups together the tuples in a relation that are collectively used by the important transactions.
- A horizontal fragment is produced by specifying a where clause condition that performs a restriction on the tuples in the relation.
- It can also be defined using the Selection operation of the relational algebra.
- Let us consider an example of horizontal fragmentation(next slide).

A sample relation instance of the relation DEPOSIT is shown in *Figure 5*.

Branch-code	Account number	Customer name	Balance
1101	3050	Suresh	5000
1101	2260	Swami	3360
1102	1170	Swami	2050
1102	4020	Khan	10000
1101	1550	Khan	620
1102	4080	Khan	1123
1102	6390	Khan	7500

Figure 5: Sample DEPOSIT relation

- Now let us decompose the table in figure 5 into horizontal fragments. Let us take branch codes 1101 and 1102.

DEPOSIT1 obtained by selection on branch-code as 1101			
Branch-code	**Account number**	**Customer name**	**Balance**
1101	3050	Suresh	5000
1101	2260	Swami	3360
1101	1550	Khan	620
DEPOSIT2 obtained by selection on branch- code as 1102			
Branch-code	**Account number**	**Customer name**	**Balance**
1102	1770	Swami	2050
1102	4020	Khan	10000
1102	4080	Khan	1123
1102	6390	Khan	7500

- The reconstruction of the relation from the fragments can be done by taking the union of all fragments.
- The previously discussed fragments can be defined in relational algebra as :

DEPOSIT1= $\sigma_{\text{branch-code=1101}}$(DEPOSIT) // Fragment 1

DEPOSIT2= $\sigma_{\text{branch-code=1102}}$(DEPOSIT) // Fragment 2

Fragment 1 is stored in the branch whose code is 1101 and fragment 2 is stored in the branch whose code is 1102.

In the above example, the two fragments are disjoint. By changing the selection predicates used to construct the fragments, we may have overlapping horizontal fragments. This is a form of data replication.

Vertical fragmentation :

- Vertical fragmentation groups together only those attributes in a relation that are used jointly by several important transactions.
- A vertical fragment is defined using the Projection operation of the relational algebra.
- Vertical fragmentation is same as that of decomposition.
- Vertical fragmentation is accomplished by adding a special attribute called a tuple-number to the schema R. A tuple number is a physical or logical address for a tuple.
- Since each tuple in R must have a unique address, the tuple number attribute is a key to the new fragments obtained.
- Let us consider an example of vertical fragmentation(next slide).

Branch-code	Account number	Customer name	Balance	Tuple-number
1101	3050	Suresh	5000	1
1101	2260	Swami	3360	2
1102	1170	Swami	2050	3
1102	4020	Khan	10000	4
1101	1550	Khan	620	5
1102	4080	Khan	1123	6
1102	6390	Khan	7500	7

Figure 7: The relation DEPOSIT of figure 5 with tuple- numbers

- The above mentioned relation can be decomposed into two fragments :

DEPOSIT3		
Branch-code	**Customer-name**	**Tuple-number**
1101	Suresh	1
1101	Swami	2
1102	Swami	3
1102	Khan	4
1101	Khan	5
1102	Khan	6
1102	Khan	7
DEPOSIT4		
Account number	**Balance**	**Tuple-number**
3050	5000	1
2260	3360	2
1170	2050	3
4020	10000	4
1550	620	5
4080	1123	6
6390	7500	7

Figure 8: Vertical fragmentation of relation DEPOSIT

- The above two fragments can be defined as :

$DEPOSIT3 = \prod_{(branch\text{-}code,\ customer\text{-}name,\ tuple\text{-}number)} (DEPOSIT)$

$DEPOSIT4 = \prod_{(account\text{-}number,\ balance,\ tuple\text{-}number)} (DEPOSIT)$

- We can reconstruct the relation from the fragments by taking natural join of the two vertical fragments. Tuple number allows direct retrieval of the tuples without the need for an index.
- Tuple numbers are system generated and should not be visible to users.

Mixed Fragmentation :

- Also known as hybrid fragmentation.
- In some cases, horizontal or vertical fragmentation of a database schema by itself is insufficient to adequately distribute the data for some applications. In this situation, hybrid fragmentation is required.
- Mixed fragmentation consists of a horizontal fragment that is vertically fragmented, or a vertical fragment that is horizontally fragmented.

Derived horizontal fragmentation :

- Some applications may involve a join of two or more relations. If the relations are stored at different locations, there may be a significant overhead in processing the join. In such cases, its more appropriate to ensure that the relations, that are joined together are at the same location. This can be achieved by using derived horizontal location.

- A stepwise methodology for distributed database design :

1. Examine the nature of distribution. Find out whether an organisation needs to have a database at each branch office , or in each city, or possibly at a regional level. This can be understood from the fragmentation. For example if database is needed at each branch office, the relations may fragment on the basis of branch number.

2. Create a detailed global e-R diagram, if so needed and identify relations from entities.

3. Analyse the most important transactions in the system and identify where horizontal or vertical fragmentation may be desirable and useful.

4. Identify the relations that are not to be fragmented. Such relations will be replicated everywhere. From the global ER diagram, remove the relations that are not going to be fragmented.

5. Examine the relations that are on one-side of a relationship and decide a suitable fragmentation schema for these relations. Relations on many-side of a relationship may be the candidates for derived fragmentation.

6. During the previous step, check for situations where either vertical or mixed fragmentation would be needed. This means we need to find the transactions which require access to a subset of attributes of a relation.

Client Server Databases

- The concept behind the Client/Server systems is concurrent, cooperative processing.
- Its an approach that presents a single systems view from a user's viewpoint.
- It involves processing on multiple, interconnected machines.
- It provides coordination of activities in a manner transparent to end users.
- Client-server database is distribution of activities into clients and a server.
- It may have a centralised or distributed database system at the server backend.
- It is a very popular commercial database implementation model.

- Emergence of Client Server Architecture
- Many relational database vendors allowed the computing to be distributed on multiple computers on network using contemporary technologies involving :
    - Low cost, high performance PCs and servers
    - Graphical User Interfaces'
    - Open Systems
    - Object-orientation
    - Workgroup computing
    - EDI and E-Mail
    - Relational Databases
    - Networking and Data Communication

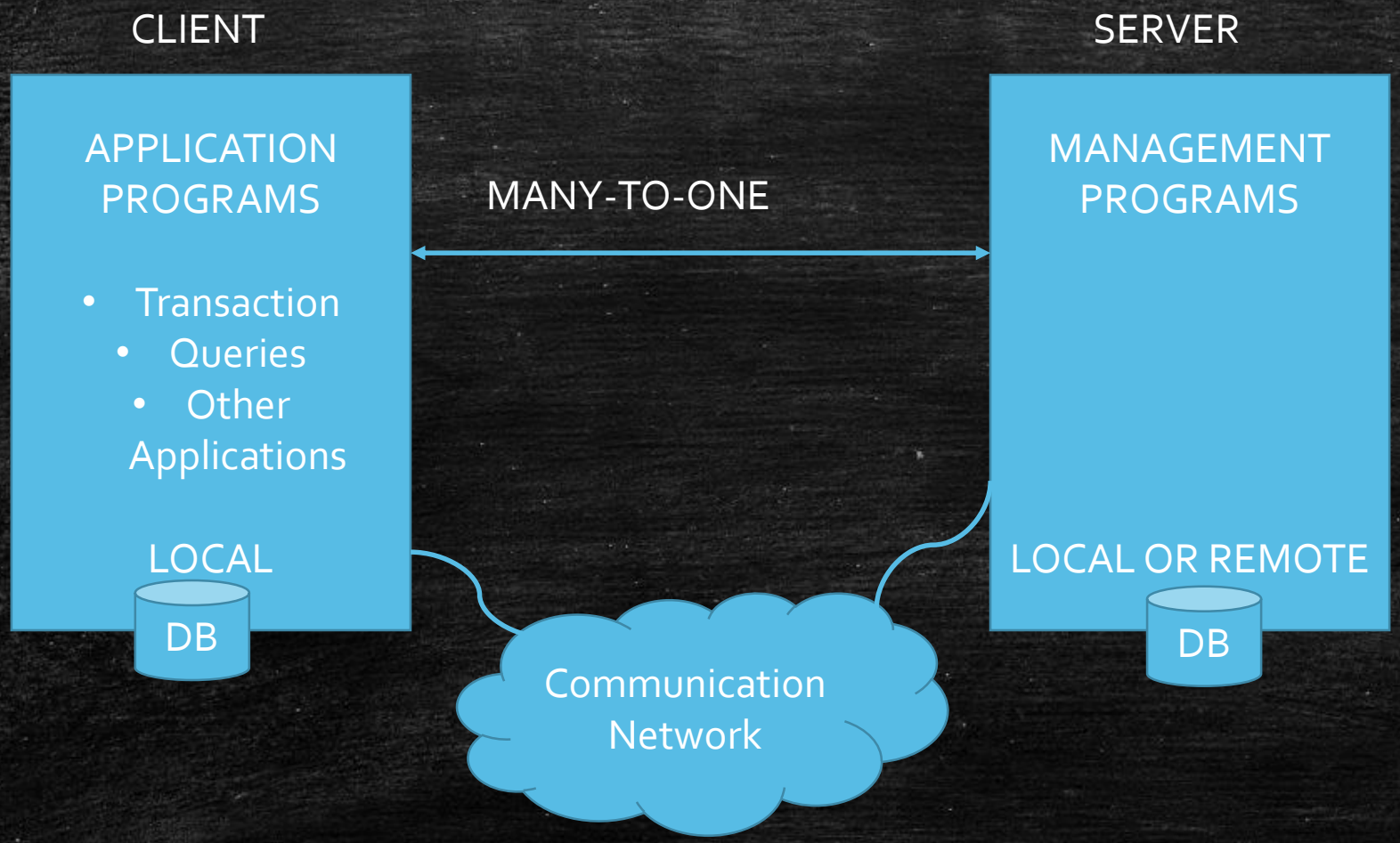
- Need for Client Server Computing
- Client / Server (C/S) architecture involves running the application on multiple machines in which each machine with its component software handles only a part of the job.
- Client machine is basically a PC or a workstation that provides presentation services and the appropriate computing, connectivity and computing services to multiple users.
- Both client machines and server machines are connected to the same network.
- As the number of users grows, client machines can be added to the network while as the load on the database server increases more servers are connected to the same network.
- Therefore, client-server combination is a scalable combination.
- Server machines are more powerful machines database services to multiple client requests.

- The client-server systems are connected through the network.
- This network can be Local Area Network or Wide Area Network.
- The client and server machines communicate through standard application program interfaces (called API) and remote procedure calls (RPC).
- The language through which RDBMS based C/S environment communicate is SQL.

- Structure of Client Server Systems
- In client/server architecture, client represents users who need services while servers provide services.
- Both client and server are a combination of hardware and software.
- Servers are separate logical objects that communicate with clients over a network to perform tasks together.
- A client makes a request for a service and receives a reply to that request.
- A server receives and processes a request, and sends back the required response.
- The client/server systems may contain two different types of architecture – 2- Tier and 3-Tier Client/Server Architectures.

- Every client/server application contains three functional units :
- Presentation logic which provides the human/machine interaction (the user interface). The presentation layer handles input from the keyboard, mouse, or other input devices and provides output in the form of screen displays. For example, the ATM machine of a bank provides such interfaces.
- Business logic is the functionality provided to an application program. For example, software that enables a customer to request to operate his or her balance on his/ her account with the bank is business logic. It includes rules for withdrawal, for minimum balance etc.. Its known as business logic as it contains business rules that drive a given enterprise.
- The bottom layer provides the generalised services needed by the other layers including file services, print services, communication services and database services. An example of such a service may be to provide the records of customer accounts.
- Theses functional units can reside on either the client or on one or more servers in the application.

A Client Server System



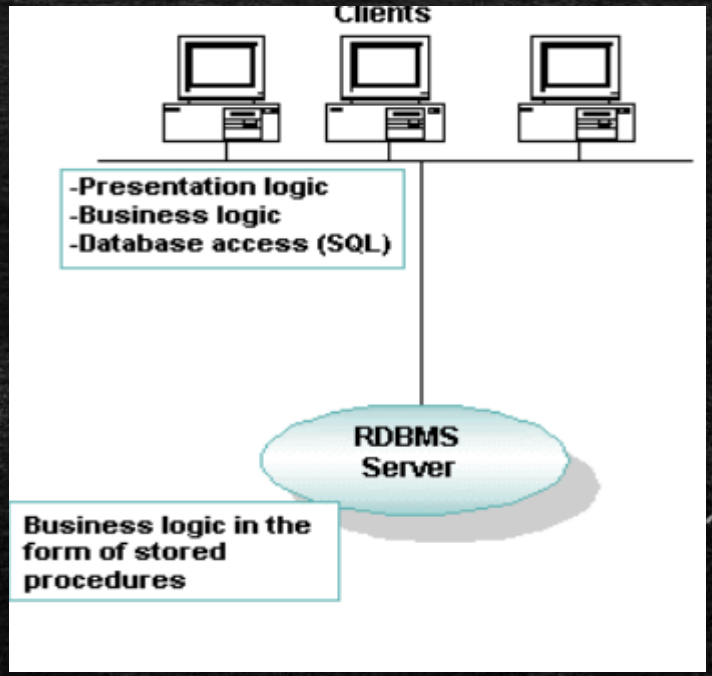
- In general two popular client/ server systems are :
- 2-tier client server Models
- 3-tier client server Models

2- Tier Client Server Models

- Initial two-tier (client/server) applications were developed to access large databases available on the server side and incorporated the rules used to manipulate the data with the user interfaces into the client application.
- The primary task of the server was simply to process as many requests for data storage and retrieval as possible.
- The two tier architecture allocates the user system interface exclusively to the client.
- It places database Management services on the server and splits the processing between client and server, creating two layers. The database management server also provides stored procedures and triggers.

- There are a number of software vendors that provide tools to simplify the development of applications for the two-tier client/server architecture.
- In 2-tier client/server applications, the business logic is put inside the user interface on the client or within the database on the server in the form of stored procedures. This results in division of the business logic between the client and server. File servers and database servers with stored procedures are examples of 2-tier architecture.
- Two tier-architecture is a good solution for distributed computing. A 2-tier client server system may have a centralised database or distributed database system at the server or servers.
- A client group may contain up to 100 people interacting simultaneously.
- A client server system does have a number of limitations. When the number of users exceeds 100, performance starts deteriorating. This is because of the server maintaining a connection via communication messages with each client, even when no work is being done.

- A second limitation of two-tier architecture is that implementation of processing management services using vendor proprietary procedures restricts flexibility and choice of DBMS for applications. The implementation of the two-tier architecture provides limited flexibility in moving program functionality from one server to another without manually regenerating procedural code.
- 2 tier client/server supports :
- Two or more operating systems
- One or more programming languages
- Local and remote databases
- Inter- application communications through networks.

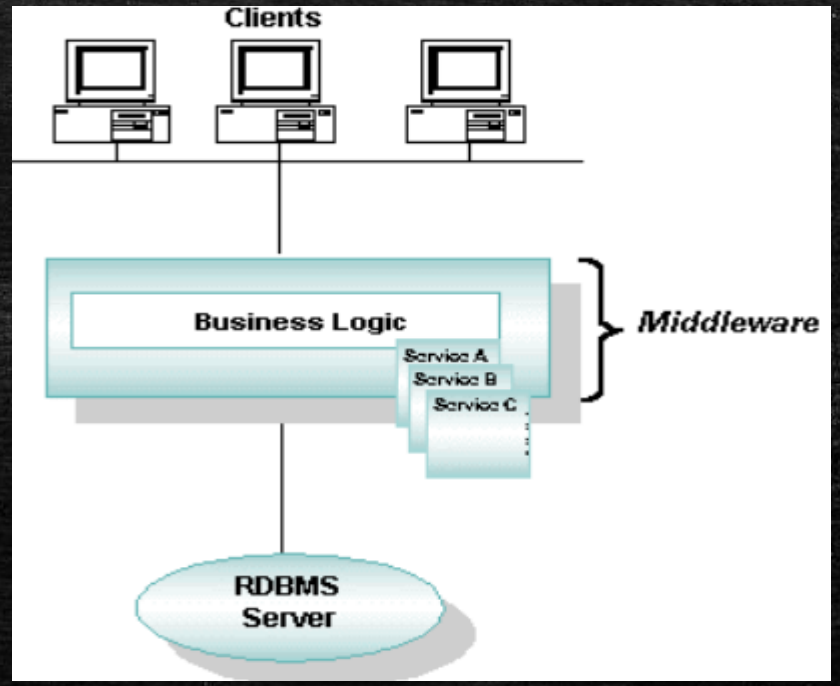


- Major functions performed by the client of a two-tier applications are :
- Present a user interface
- Gather and process user interface
- Perform the requested processing
- Report the status of the request
- This sequence of command can be repeated as many times as necessary, because servers provide only access to the data, the client uses its local resources to perform most of the processing.
- The client application must contain information about where the data resides and how it is organised in the server database.
- Once the data is retrieved, the client is responsible for formatting and display it to the user.

3-tier architecture

- The three tier architecture emerged to overcome the limitations of the two tier architecture.
- In three tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. The middle tier may consist of transaction processing monitors, message servers, or application servers. It can perform queuing, application execution and the database staging.
- For example, on a middle tier that provides queuing, the client can deliver its request to the middle layer and simply gets disconnected because the middle layer adds scheduling and prioritisation for various tasks that are currently being performed.
- The 3-tier client/server architecture has improved performance for client groups with a large number of users (in thousands) and improves flexibility when compared to the two-tier approach. Flexibility is in terms of simply moving an application on to different computers in three-tier architecture. It has become as simple as "drag and drop" tool.

- Mainframes have found a new use as server in three-tier architectures.
- In 3-tier client/server applications, the business logic resides in the middle tier, separate from the data and user interface. In this way, processes can be managed and deployed separately from the user interface and the database. Also, 3-tier systems can integrate data from multiple sources.
- It supports :
- Multiple operating systems
- One or more programming languages
- Local and remote databases
- Inter-application communication through network.
- Message routing



Advantages Of Client Server Computing

- They provide low cost and user friendly environment.
- They offer expandability that ensures that the performance degradation is not so much with increased load.
- They allow connectivity with the heterogeneous machines deals with real time data feeders like ATMs, Numerical machines.
- They allow enforcement of database management activities including security, performance, backup, integrity to be a part of the database server machine. Thus , avoiding the requirement to write a large number of redundant piece of code dealing with database field validation and referential integrity.
- By allowing multiple users to simultaneously access the same application data, updates from one computer are instantly made available to all computers that had access to the server.

Client/Server systems can be used to develop highly complex multi-user database applications being handled by any mainframe computer.

- Since PCs can be used as clients, the application can be connected to the spreadsheets and other applications through Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE). If the load on database machine grows, the same application can be run on a slightly upgraded machine provided it offers the same version of RDBMSs.