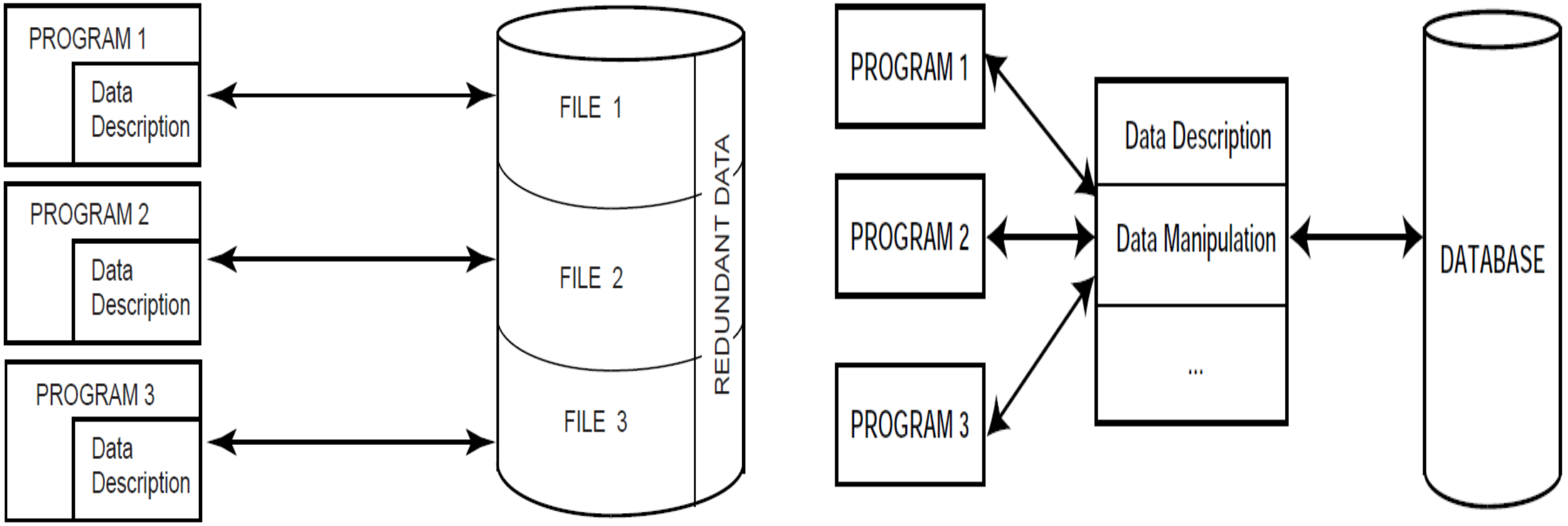
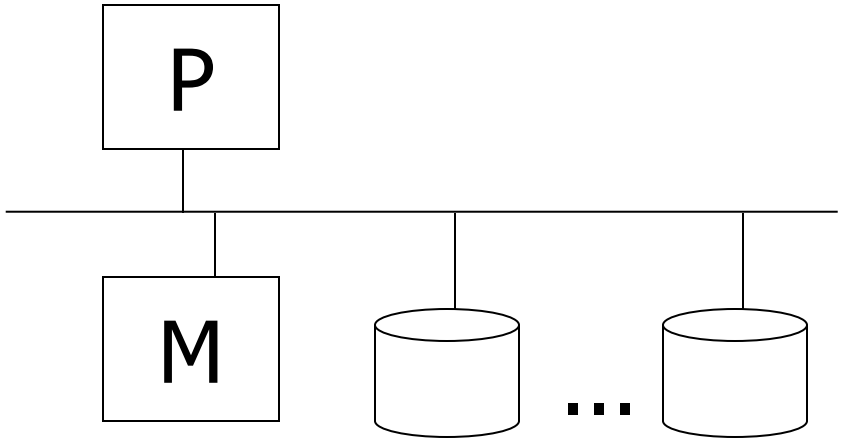
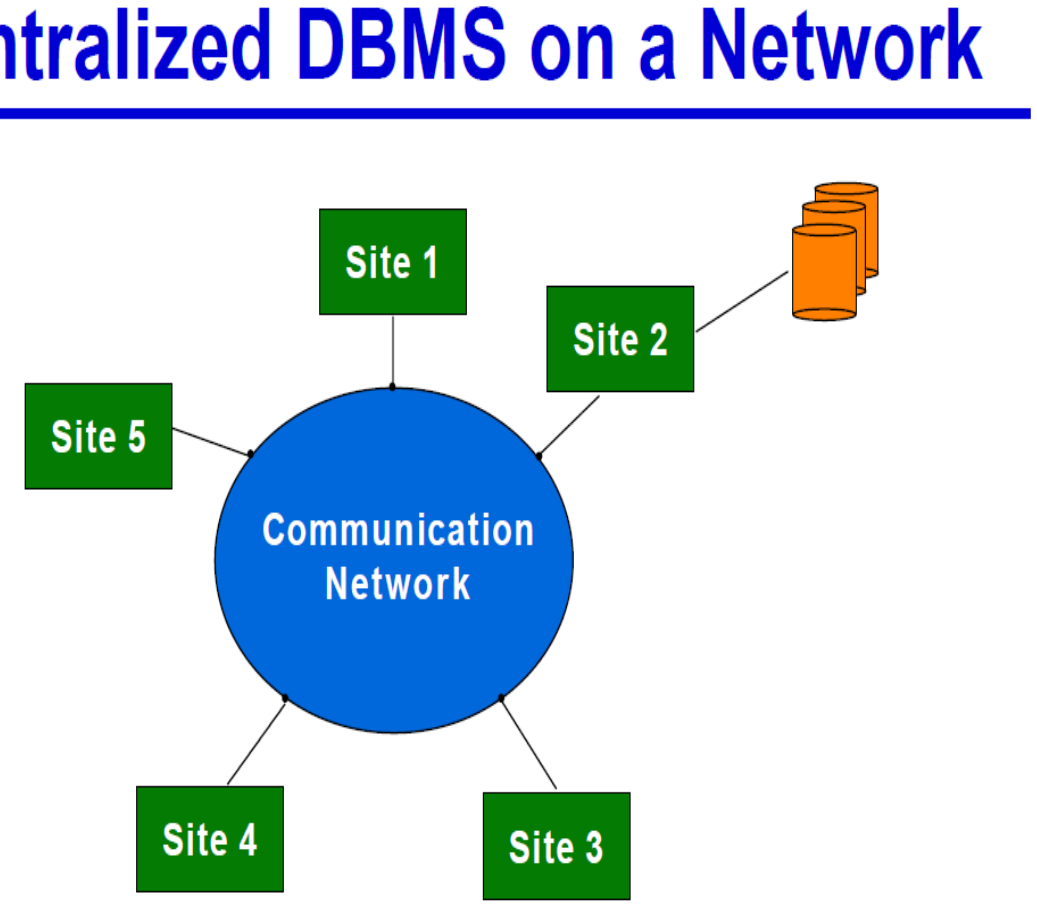
NEED FOR DDBMS

- A centralized database management system (DBMS) is a complex software program that allows an enterprise to control its data on a single machine.
- In centralized DBMS operational data of an enterprise are integrated and hence we have centralized controlled access to the data.
- The computer network technology goes against all types of centralization efforts. So the focus will be on integration rather than centralization.



Centralized DB systems





- Simplifications:
  - single front end
  - one place to keep data, locks
  - if processor fails, system fails, ...

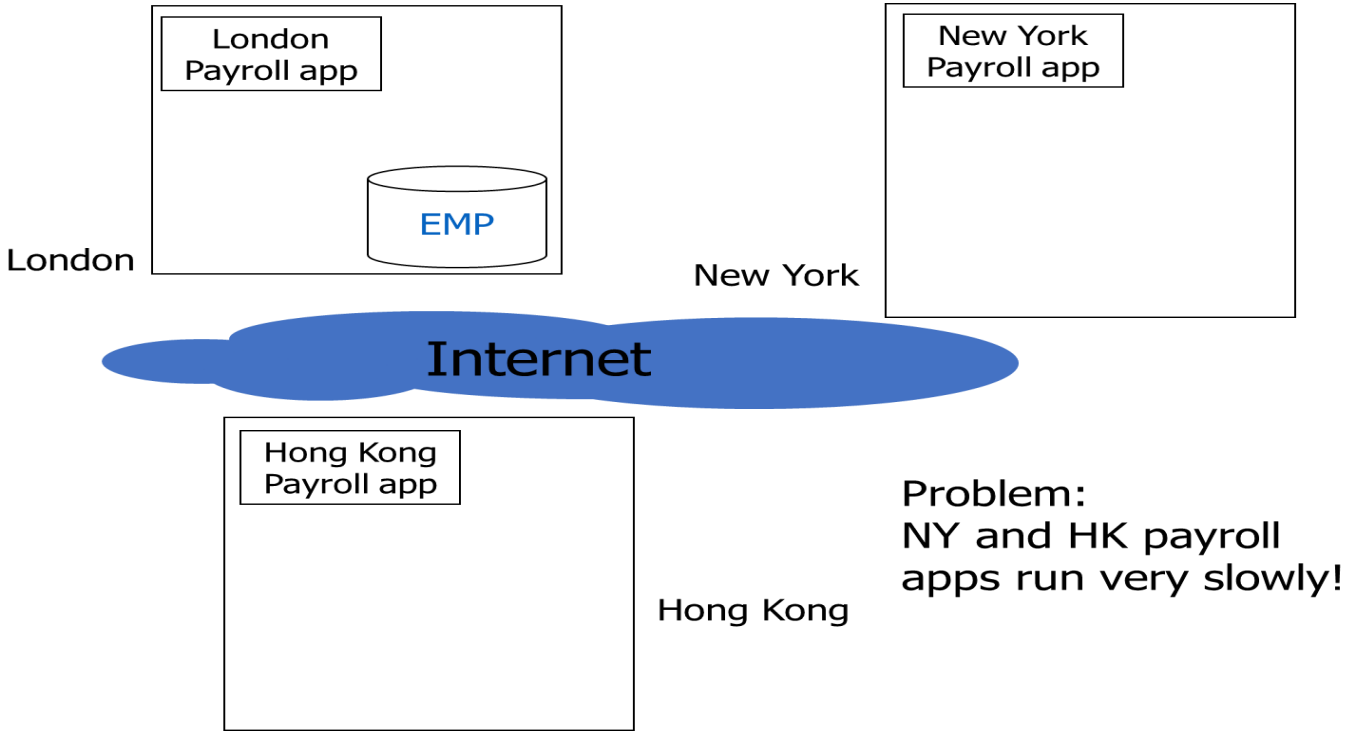
Moving over to DDBMS

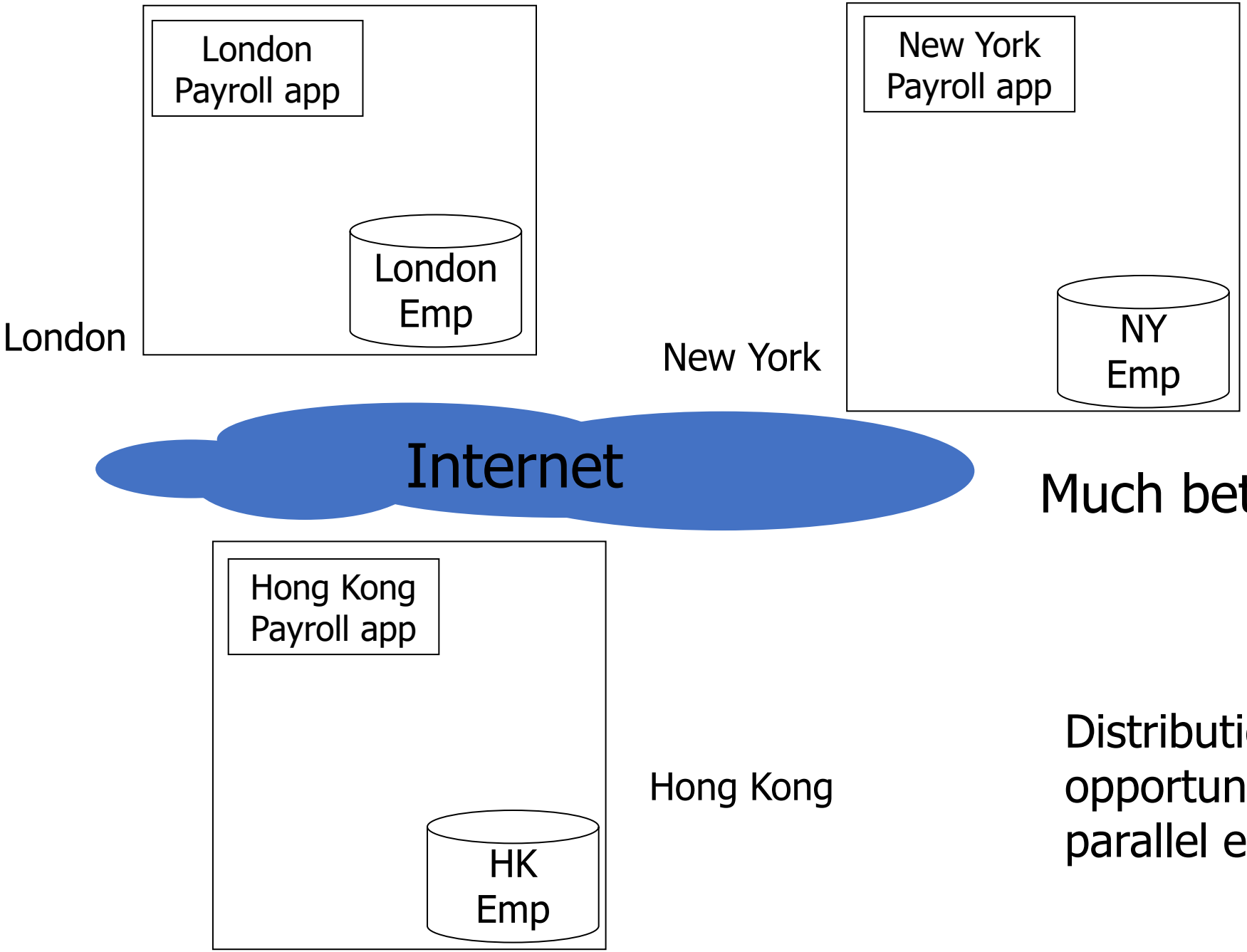
- **Distributed Nature of Organizational Units**: Most organizations in current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data**: Multiple organizational units often need to communicate with each other and share their data and resources. It 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**: Replication of data automatically helps in data recovery. 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.

Case Study: Distributed Database

- Example: IBM has offices in London, New York, and Hong Kong.
- Employee data:
  - EMP(ENO, NAME, TITLE, SALARY, …)
- Where should the employee data table reside?

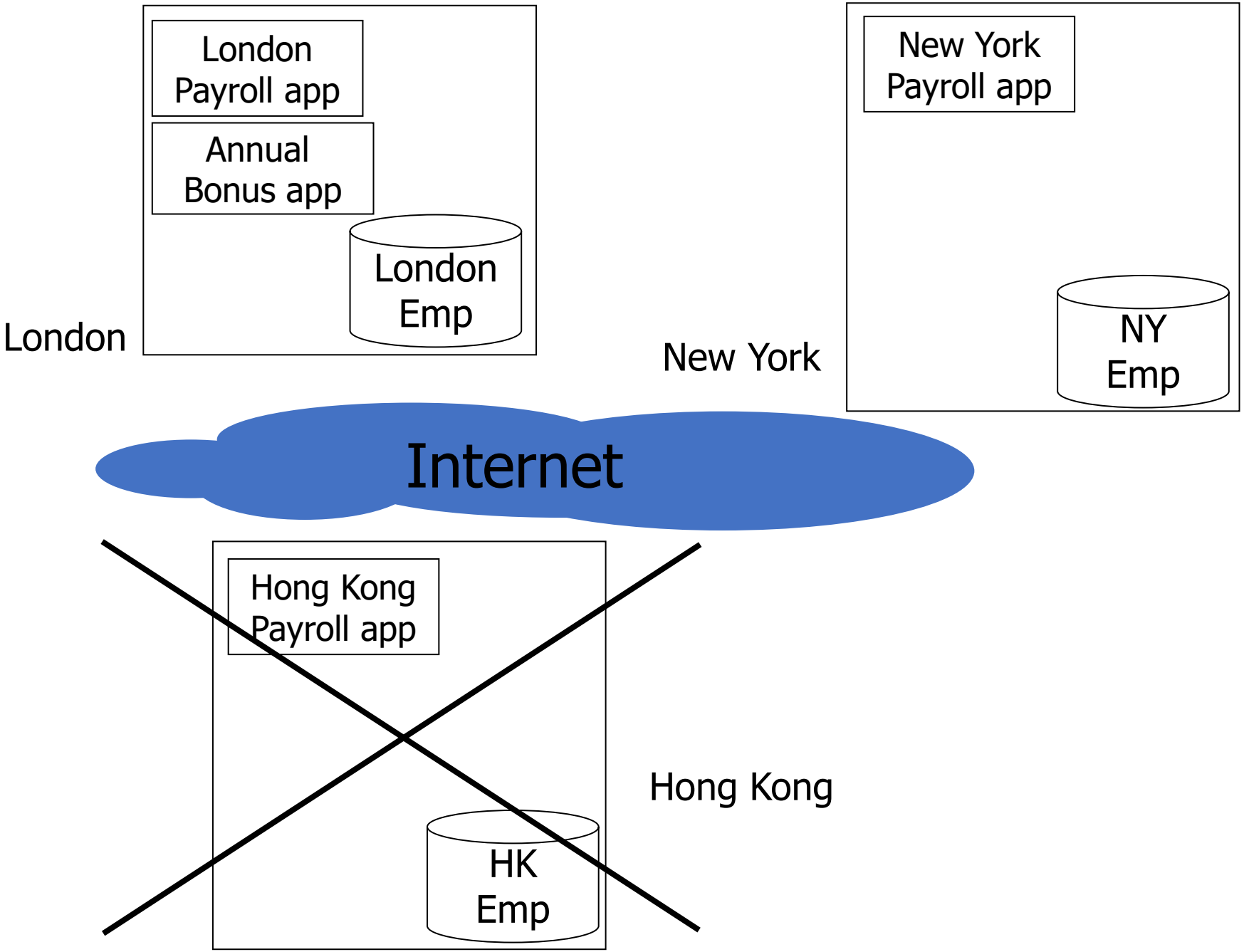
- Mostly, employee data is managed at the office where the employee works
- Periodically, IBM needs consolidated access to employee data
  - E.g., IBM changes benefit plans and that affects all employees.

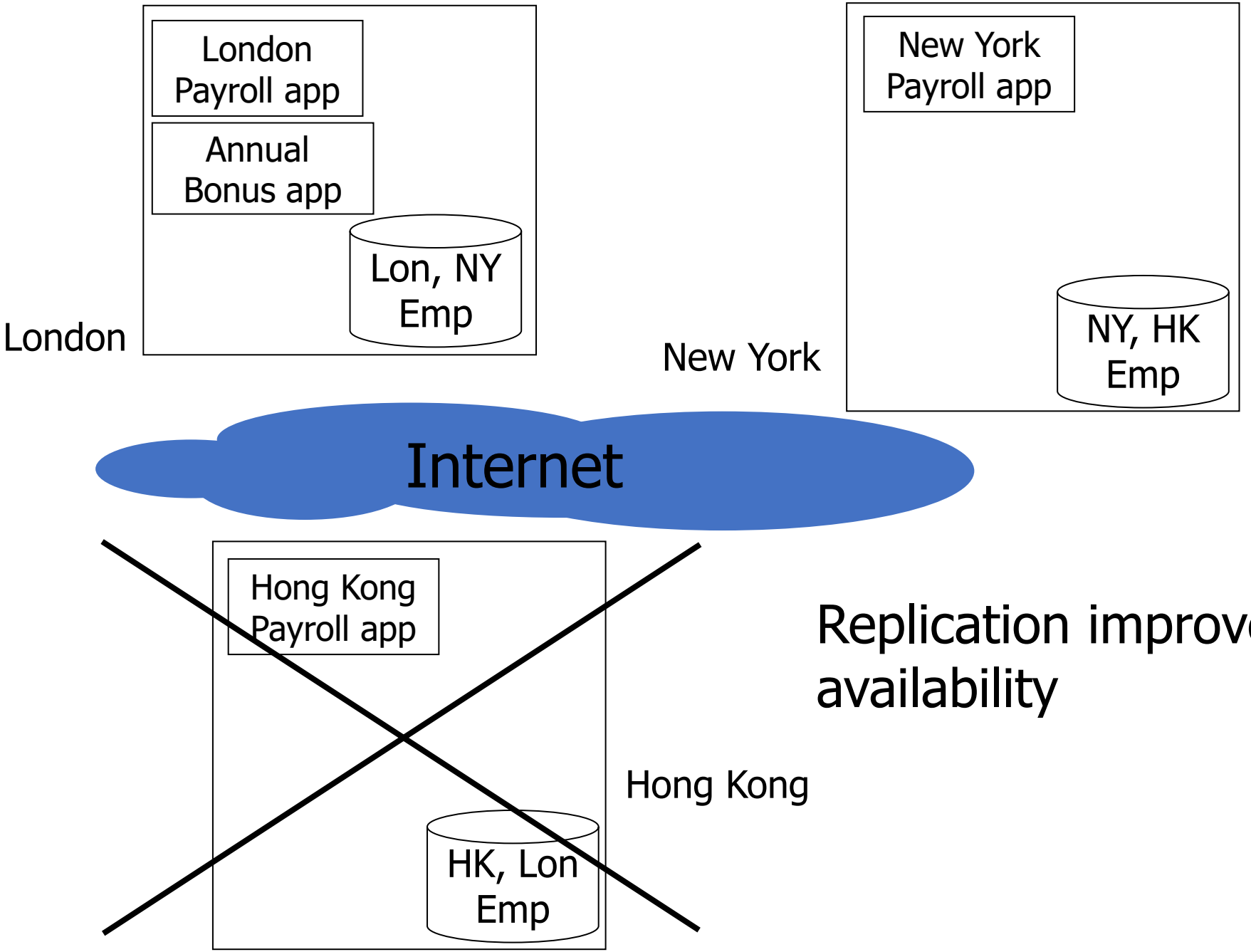




Much better!!

Distribution provides opportunities for parallel execution

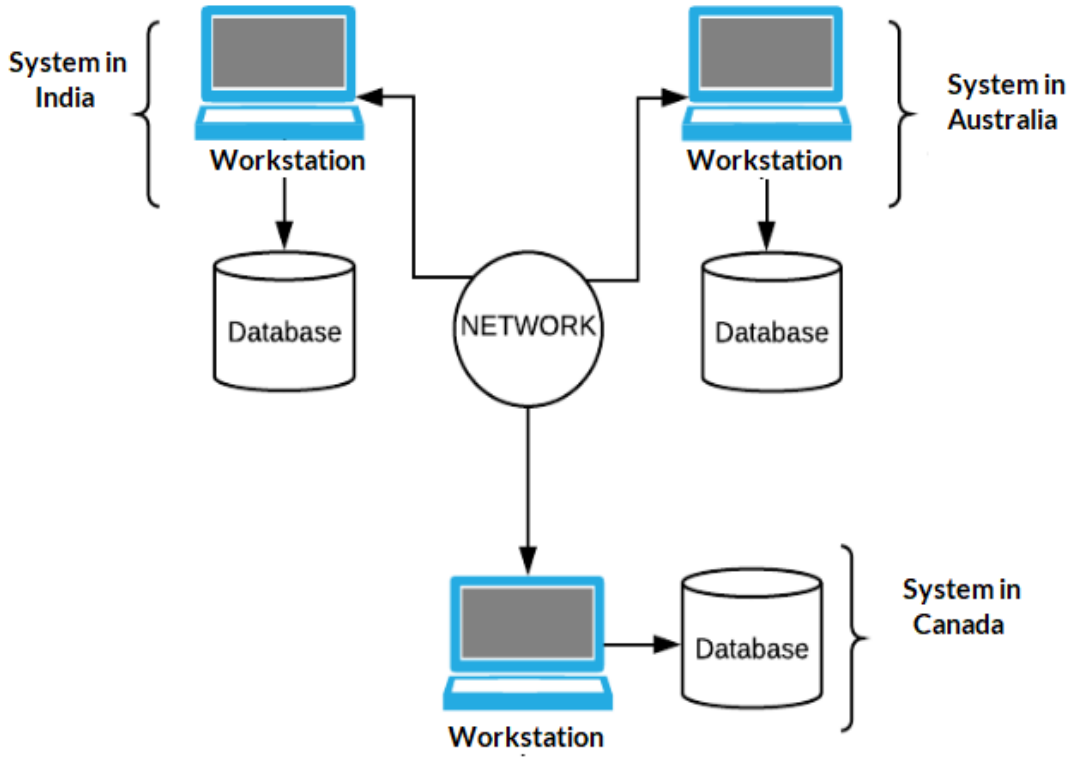

London
Payroll app
Annual
Bonus app
London
Emp
London
New York
Payroll app
NY
Emp
New York
Internet
Hong Kong
Payroll app
HK
Emp
Hong Kong



Replication improves availability

Idea of Distributed Database (DDB) System

The database is not stored at a single location. Rather, it may be stored in multiple computers at the same place or geographically spread far away. Despite all this, the distributed database appears as a single database to the user
So, DDB is, logically interrelated collection of shared data, physically distributed over a computer network and data need to be structured.



As seen in the figure, the components of the distributed database can be in multiple locations such as India, Canada, Australia, etc. However, this is transparent to the user i.e the database appears as a single entity.

Idea of Distributed Database (DDB) System contd..

A DDB is a collection of data which belong logically to the same system, but are spread over the sites of a computer network. Here two important points are

1. **Distribution –** implies the whole data not residing at the same site
2. **Logical correlation-** data have some properties which tie them together

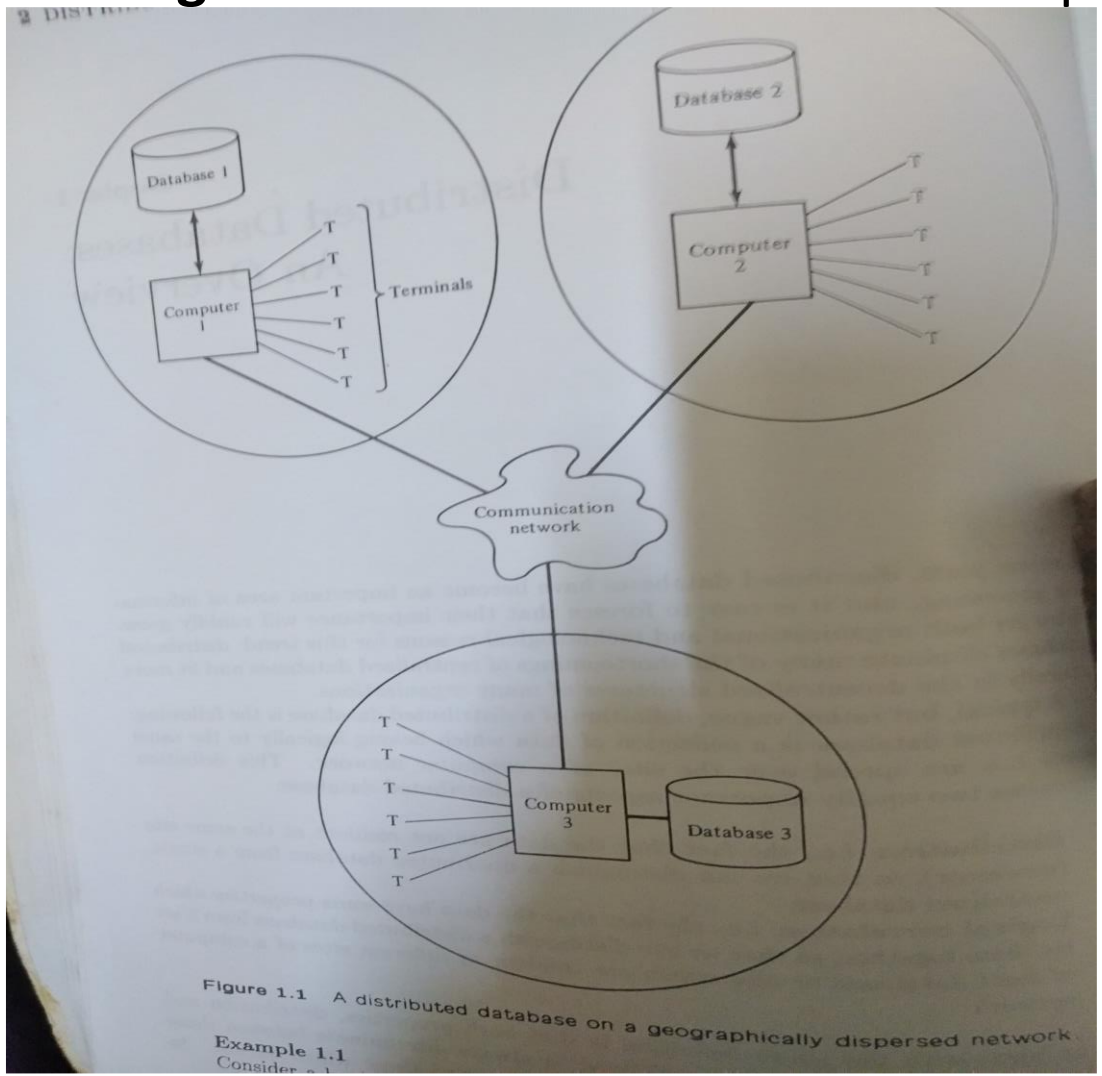


Figure 1.1 A distributed database on a geographically dispersed network.

Example 1.1
Consider a

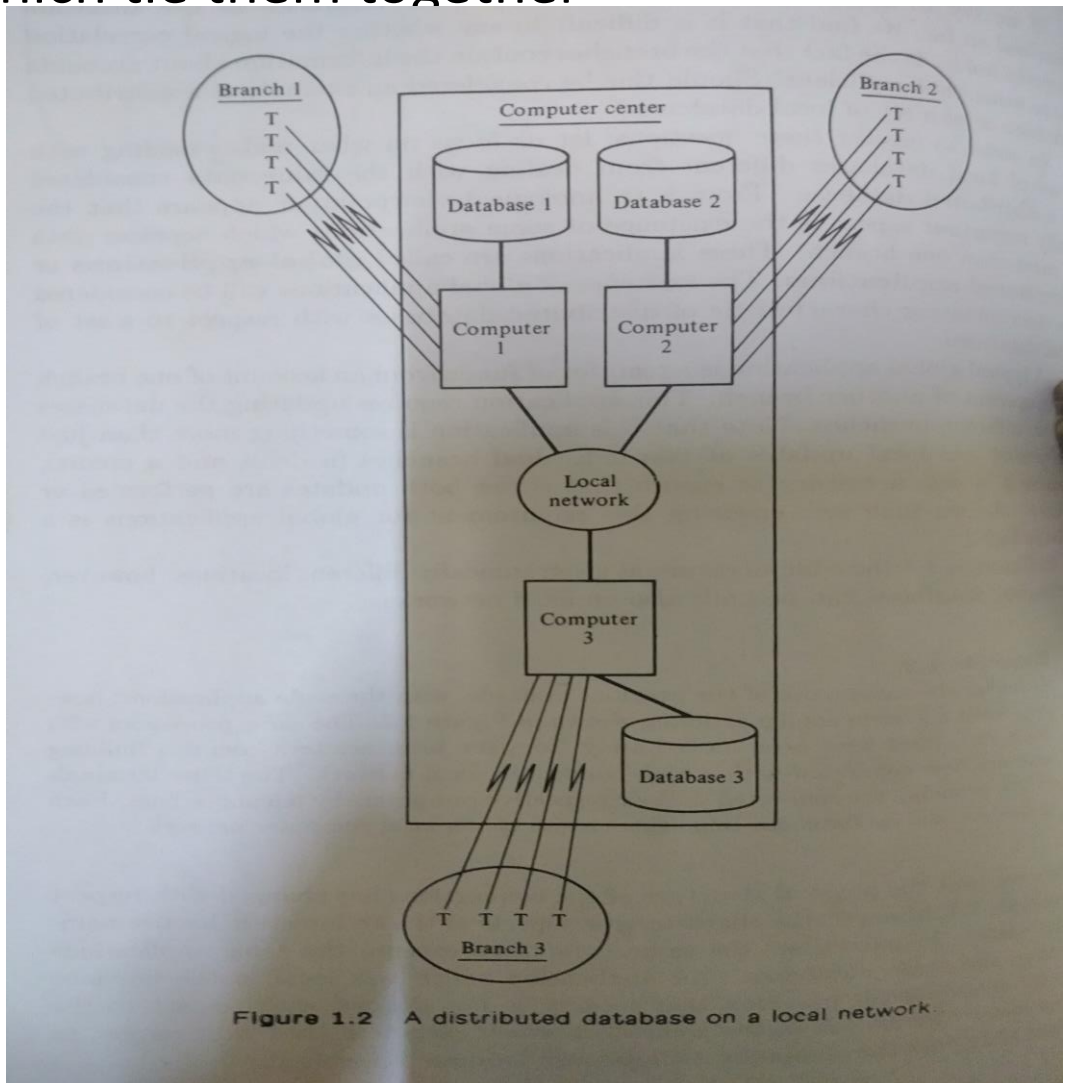


Figure 1.2 A distributed database on a local network.

What is not Considered as Distributed Database (DDB) System contd..

- The data of different branches are distributed on three backend computers, which perform database management functions.
- The application programs are executed by a different computer, which requests database access services from the backend when necessary.
- Here data distribution is not relevant from the application viewpoint and existence of local application is missing

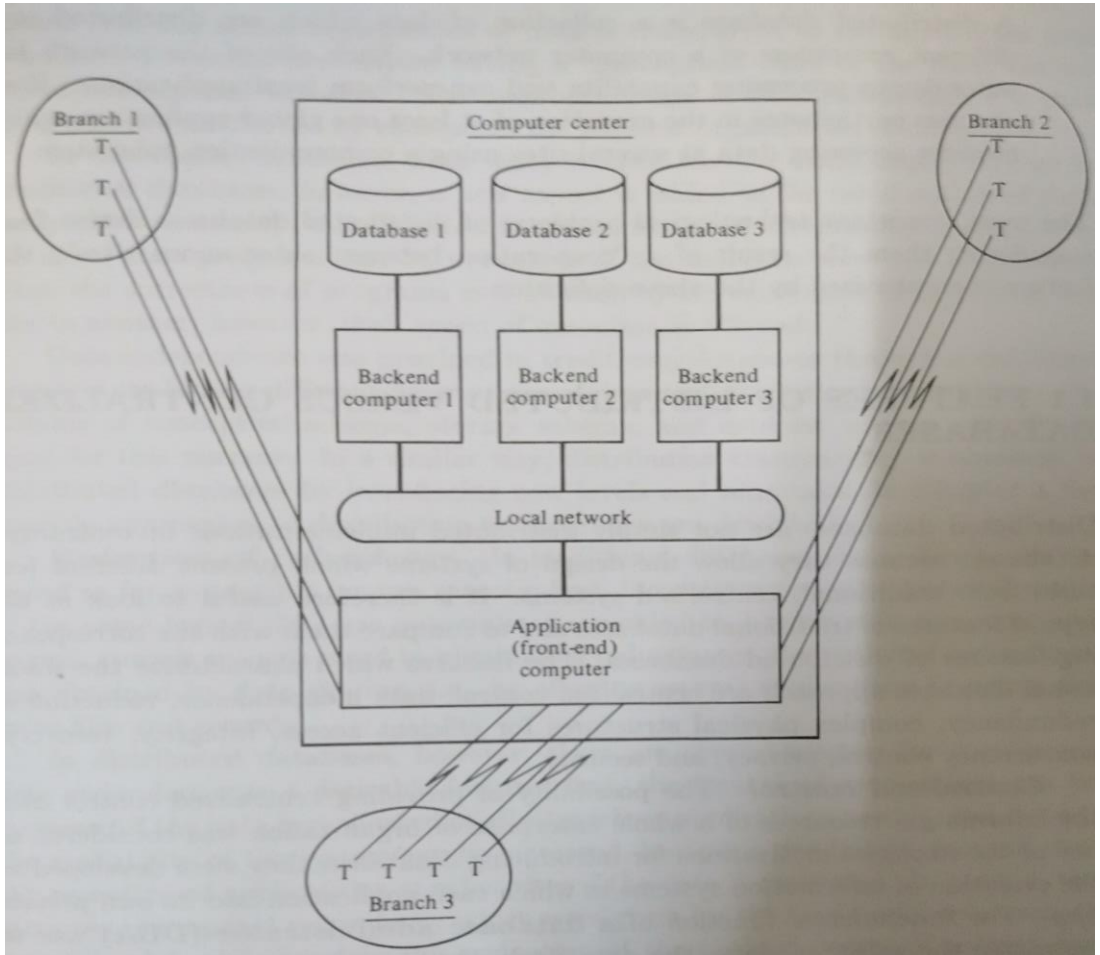


Figure 1.3 A multiprocessor system.

Example 1.3

Consider the same bank of the previous example but with the system configuration shown in Figure 1.3. The data of the different branches are distributed on three "backend" computers, which perform the database management functions. The application programs are executed by a different computer, which requests database access services from the backends when necessary.

The reason for not considering this type of system a distributed database is

- **DDBMS in a Nutshell**

- DDBMS is a set of multiple interconnected databases that is distributed over the computer network or internet.
- DDBMS manages these distributed databases and provides mechanisms so as to make these databases transparent to the users i.e it appears as one single database to users.
- In these systems, data is intentionally distributed among multiple nodes so that all computing resources of the organization can be optimally used.
- 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.
- DDBMS is not a loosely connected file system.

MAIN OPERATIONS OF DDBMS

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and ensures that 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.

3.1 Fundamentals of Distributed Databases

In recent years, the distributed database system has been emerging as an important area of information processing, and its popularity is increasing rapidly. A **distributed database** is a database that is under the control of a central DBMS in which not all storage devices are attached to a common CPU. It may be stored on multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Collections of data (e.g., in a database) can be distributed across multiple physical locations. In short, a **distributed database** is a logically interrelated collection of shared data, and a description of this data is physically distributed over a computer network. A distributed database must ensure the following:

- **The distribution is transparent** – users must be able to interact with the system as if it is a single logical system. This applies to the system performance and method of accessing amongst other things.
- **The transactions are transparent** – each transaction must maintain database integrity across multiple databases. Transactions may also be divided into subtransactions; each subtransaction affects one database system.

29°C AQI 52 ENG 02:09 29-07-2021

A distributed database management system (DDBMS) can be defined as follows.

A DDBMS consists of a single logical database that is split into a number of **partitions** or **fragments**. Each partition is stored on one or more computers under the control of a separate DBMS,

Copyrighted material

with the computers connected by a communication network. Each site may have a substantial degree of independence. Each site in the distributed system is capable of independently processing user requests that require access to local data as well as it is capable of processing user requests that require access to remote data stored on other computers in the network. The sites in a distributed system have agreed to work together so that a user can access any data from anywhere in the network exactly as if the data are stored at the user's own site. A **distributed database system** allows applications to access various data items from local and remote databases. Applications are classified into two categories depending on whether the transactions access data from local site or remote sites.

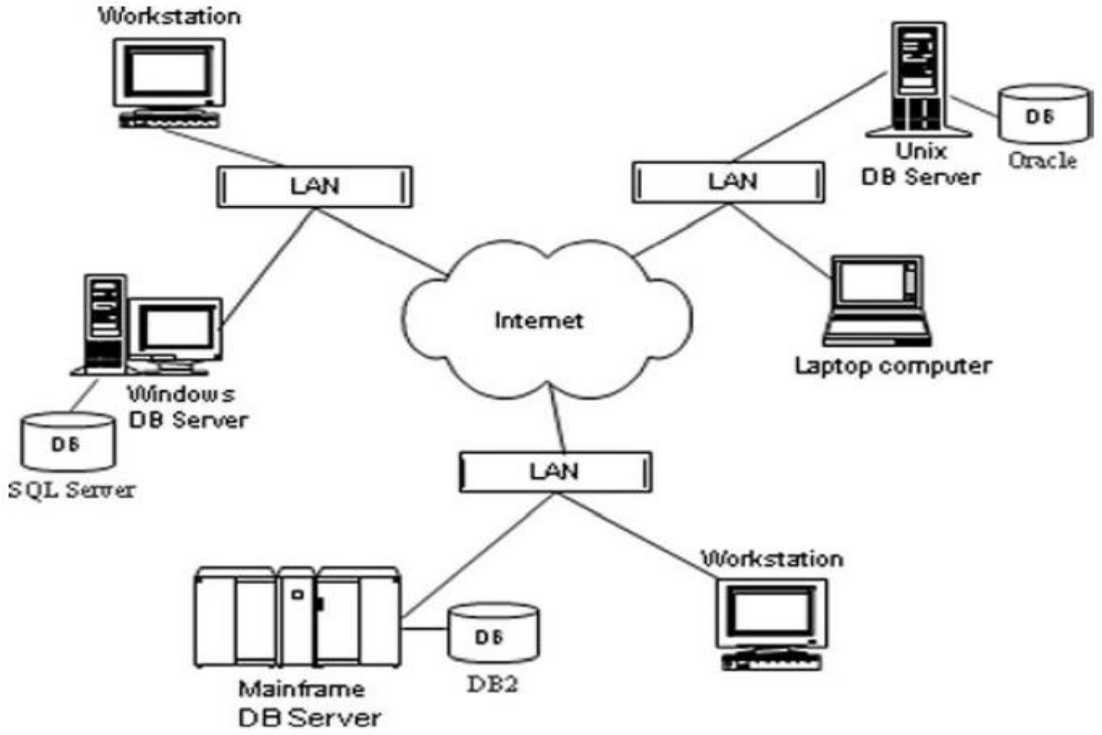
- **Local applications** – These applications require access to local data only and do not require data from more than one site.
- **Global applications** – These applications require access to data from other remote sites in the distributed system.

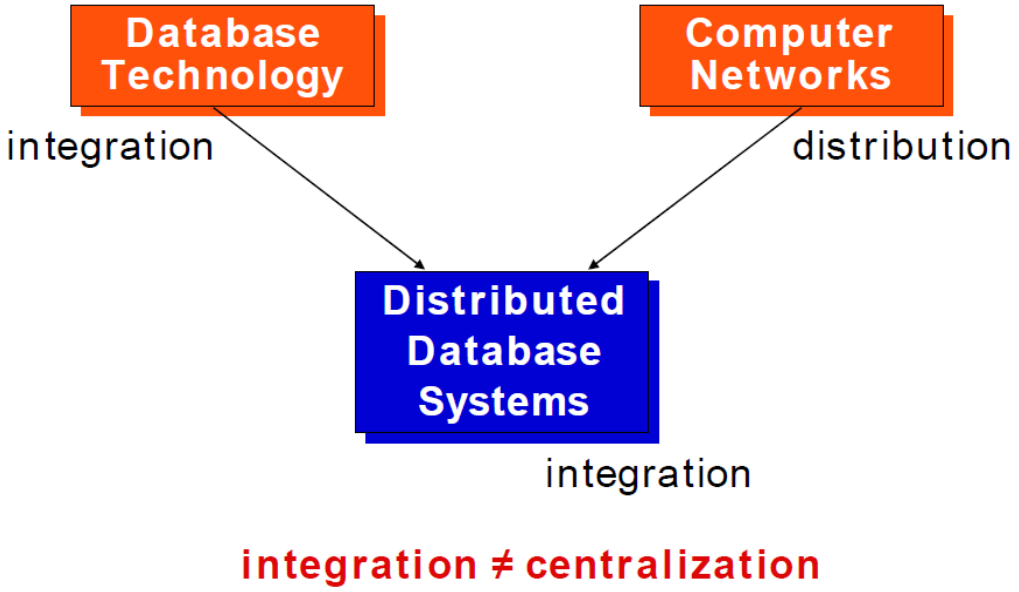
Overview of DDB

- **Distributed DBMS (DDBMS):** Software system that permits the management of the distributed database and makes the distribution transparent to users
- **DDBMS** Governs storage and processing of logically related data over interconnected computer systems in which both data and processing functions are distributed among several sites. So DDBS -> DDB + D-DBMS
- Sharing of data can be achieved by developing a distributed database system which makes the data accessible by all sites and stores the data close to the site where it is most frequently used.
- **Distributed databases** utilize multiple nodes. More nodes in the system provide more computing power, offer greater availability, and resolve the single point of failure issue.
- DDBMS ensures that updates in one database automatically reflected on other databases (as applicable) in different sites.

POSSIBLE CASE OF DDBMS

- IBM's DB2 running on IBM mainframe server, contains information about payroll and human resources.
- Oracle's DBMS maintains project information and Microsoft's SQL Server keeps track of production information.
- If we need to combine the information stored in different types of DBMS, we can adopt two approaches-.
- **In manual approach**, we would fetch data through queries form each individual server DB and then transfer results from each DB to the personal workstation to merge them.
- **For automated approach**, the company needs to combine all the three Centralized DBs into a new, single, distributed DB that needs to be implemented. Here the users will have the illusion that all the combined data content is stored and controlled locally by one system. The DDBEs query processor handles all the necessary coordination, execution, and merging automatically.





3.4 An Example of Distributed DBMS

Assume that an IT company has a number of branches in different cities throughout the country. Each branch has its own local system that maintains information regarding all the clients, projects and employees in that particular branch. Each such individual branch is termed a site or a node. All sites in the system are connected via a communication network. Each site maintains three relational schema: Project for project information, Client for client information and Employee for employee information, which are listed in the following.

Project = (project-id, project-name, project-type, project-leader-id, branch-no, amount)

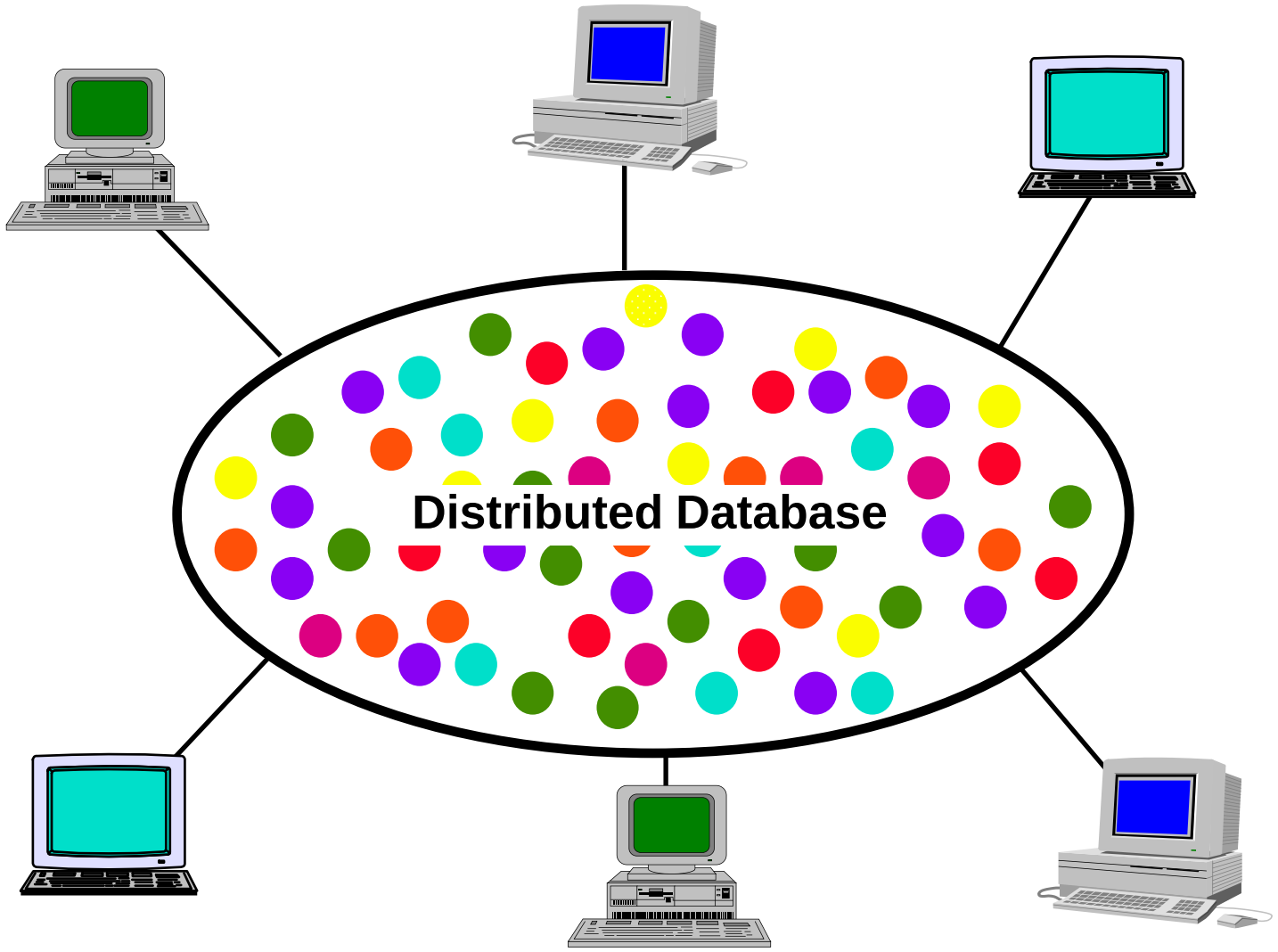
Client = (client-id, client-name, client-city, project-id)

Employee = (emp-id, emp-name, designation, salary, emp-branch, project-no)

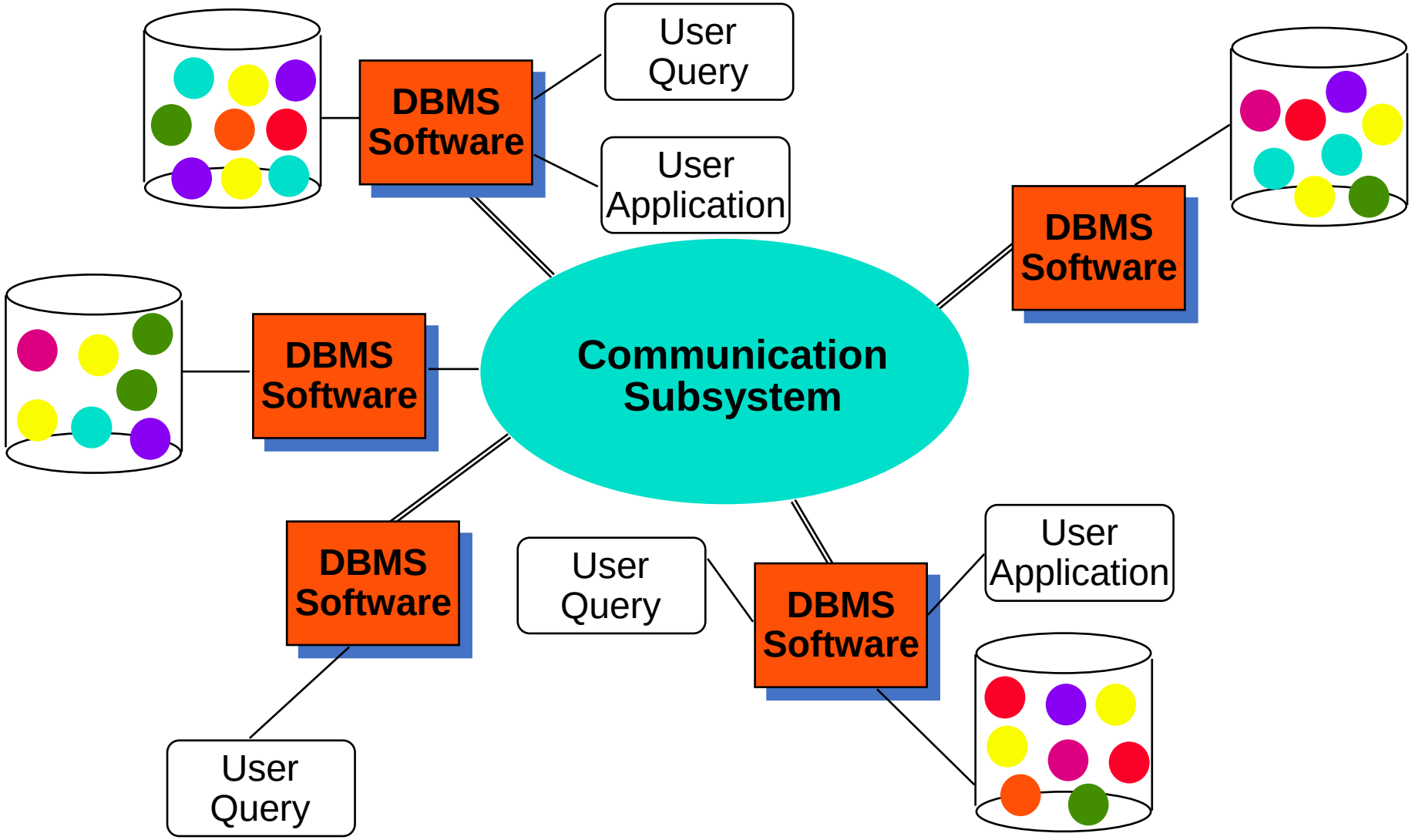
Here the underlined attributes represent the primary keys for the corresponding relations. There also exists one single site that maintains the information about all branches of the company. The single site containing information about all the clients, projects and employees in all branches of the company maintains an additional relational schema Branch, which is defined as follows:

Branch = (branch-no, branch-name, branch-city, no-of-projects, total-revenue)

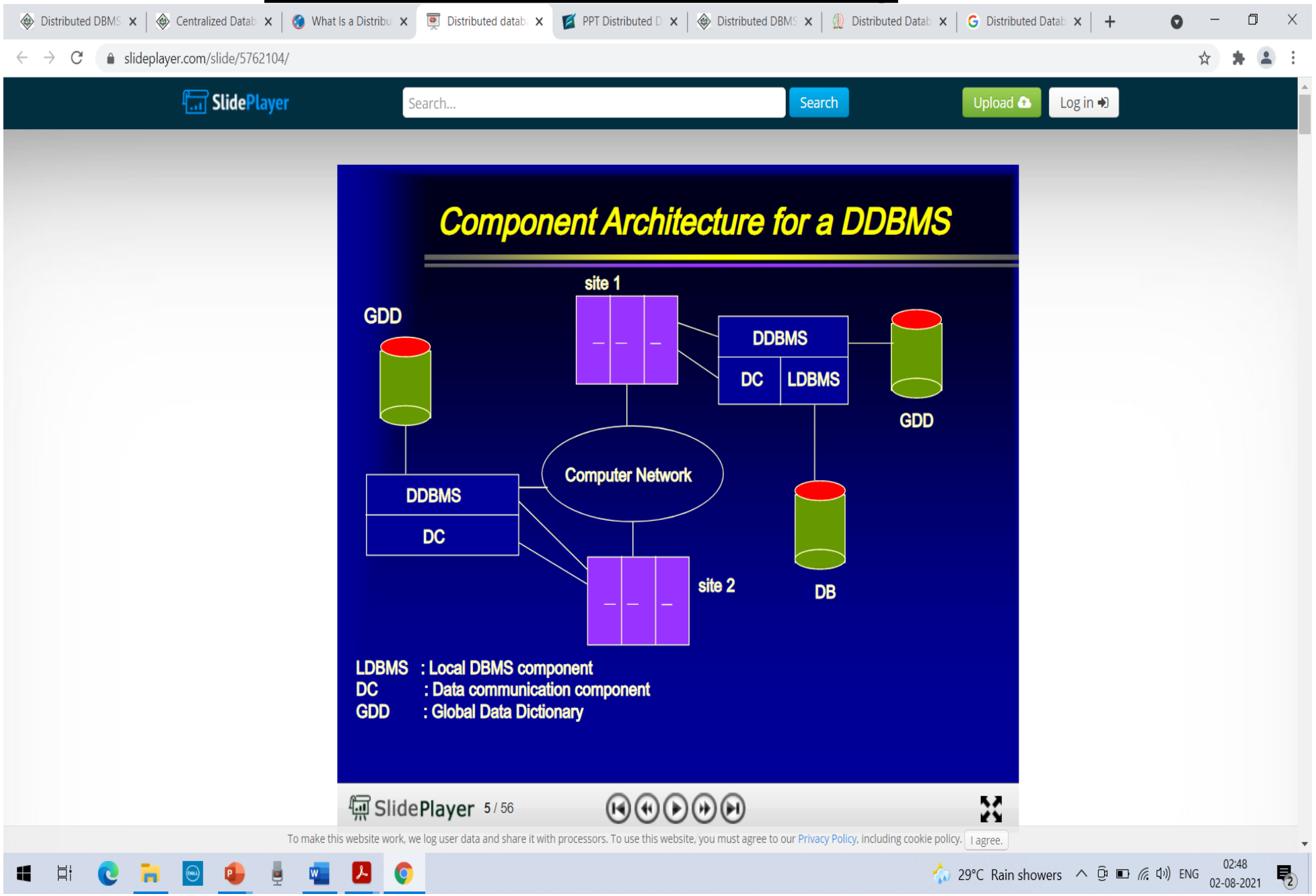
Distributed Database - User View



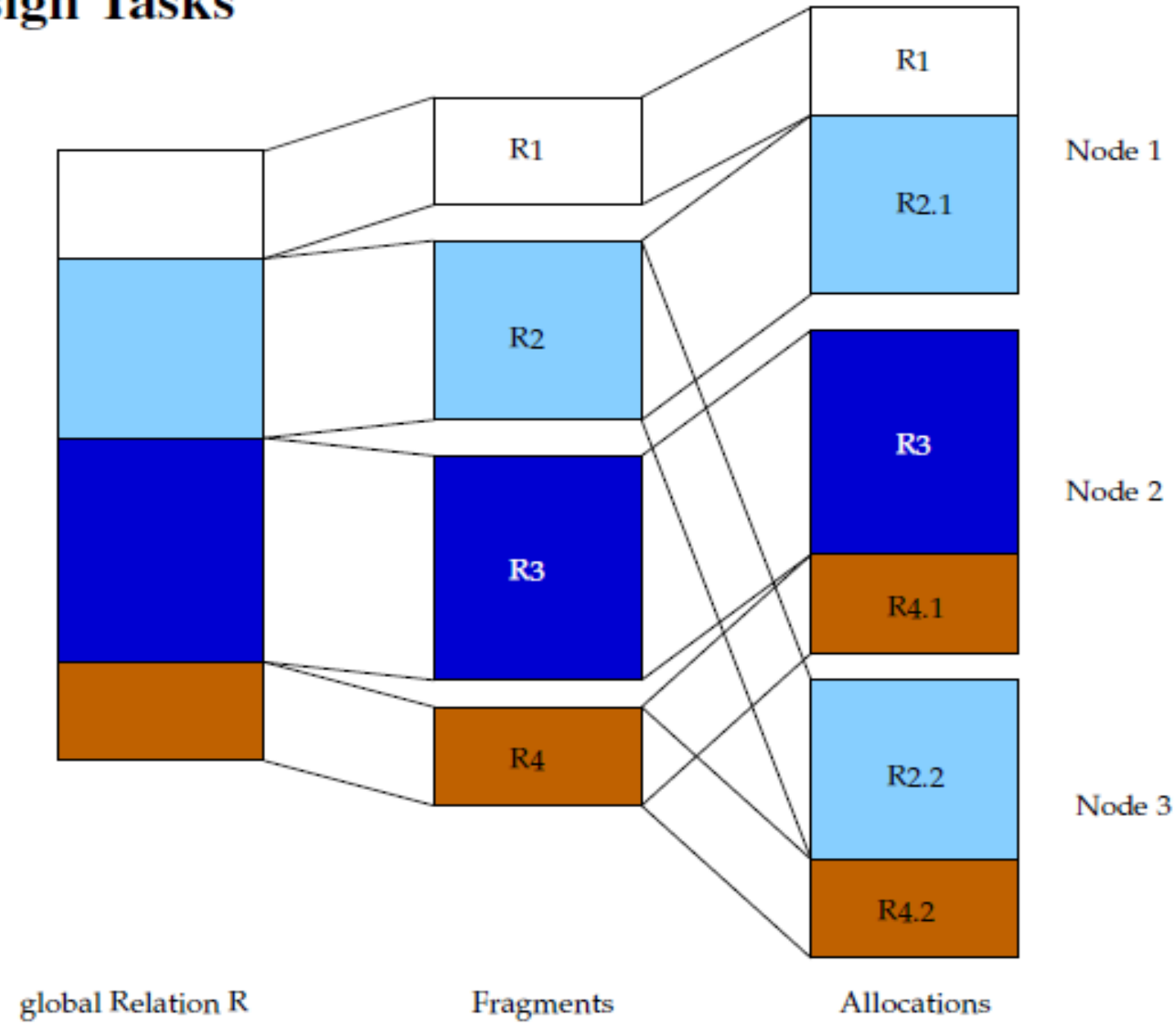
Distributed DBMS - Reality



Distributed DBMS - Reality



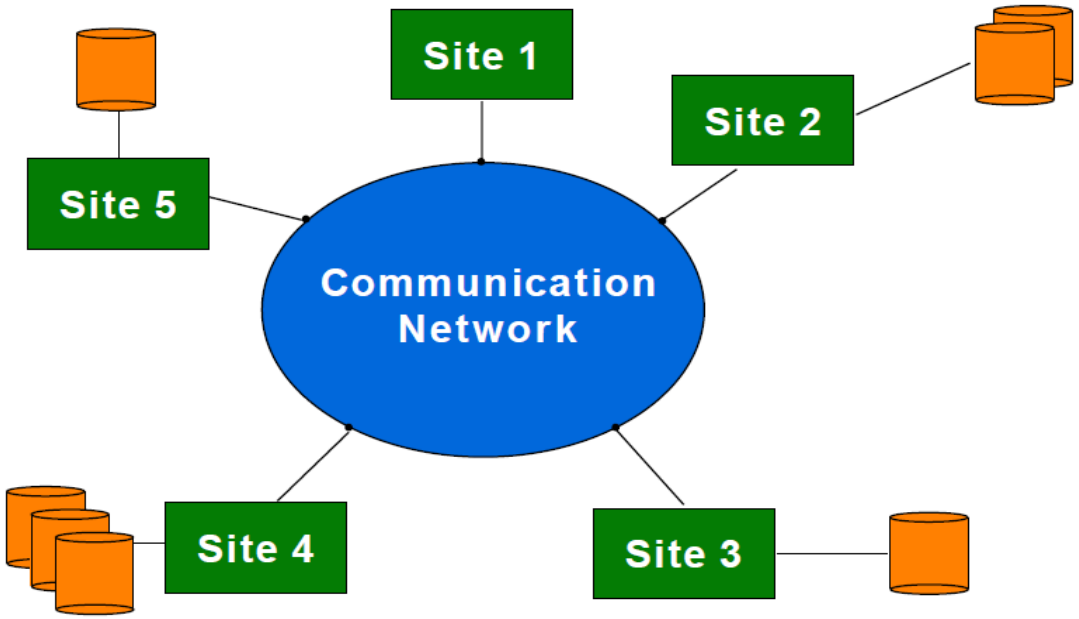
Distribution Design Tasks

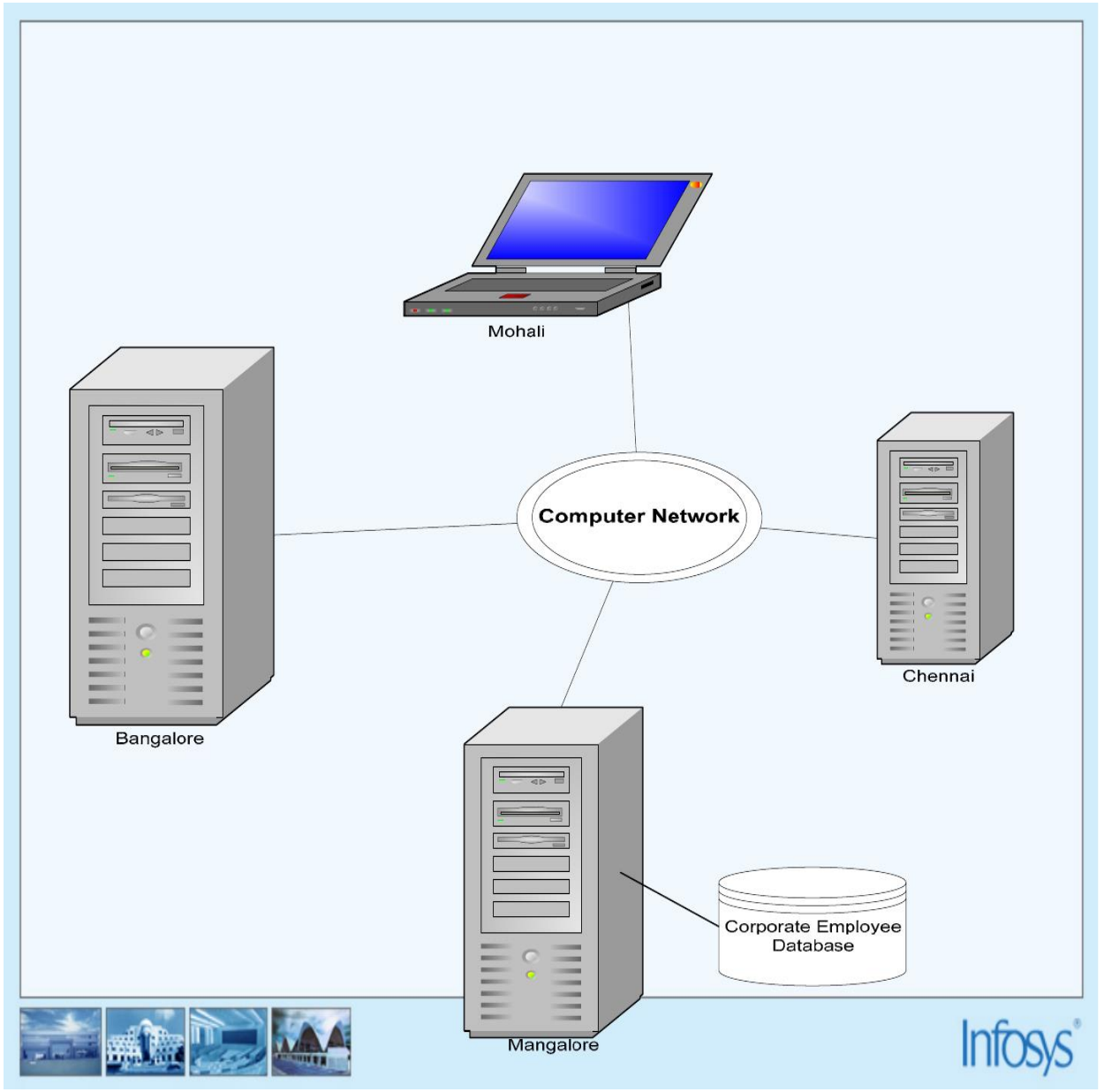


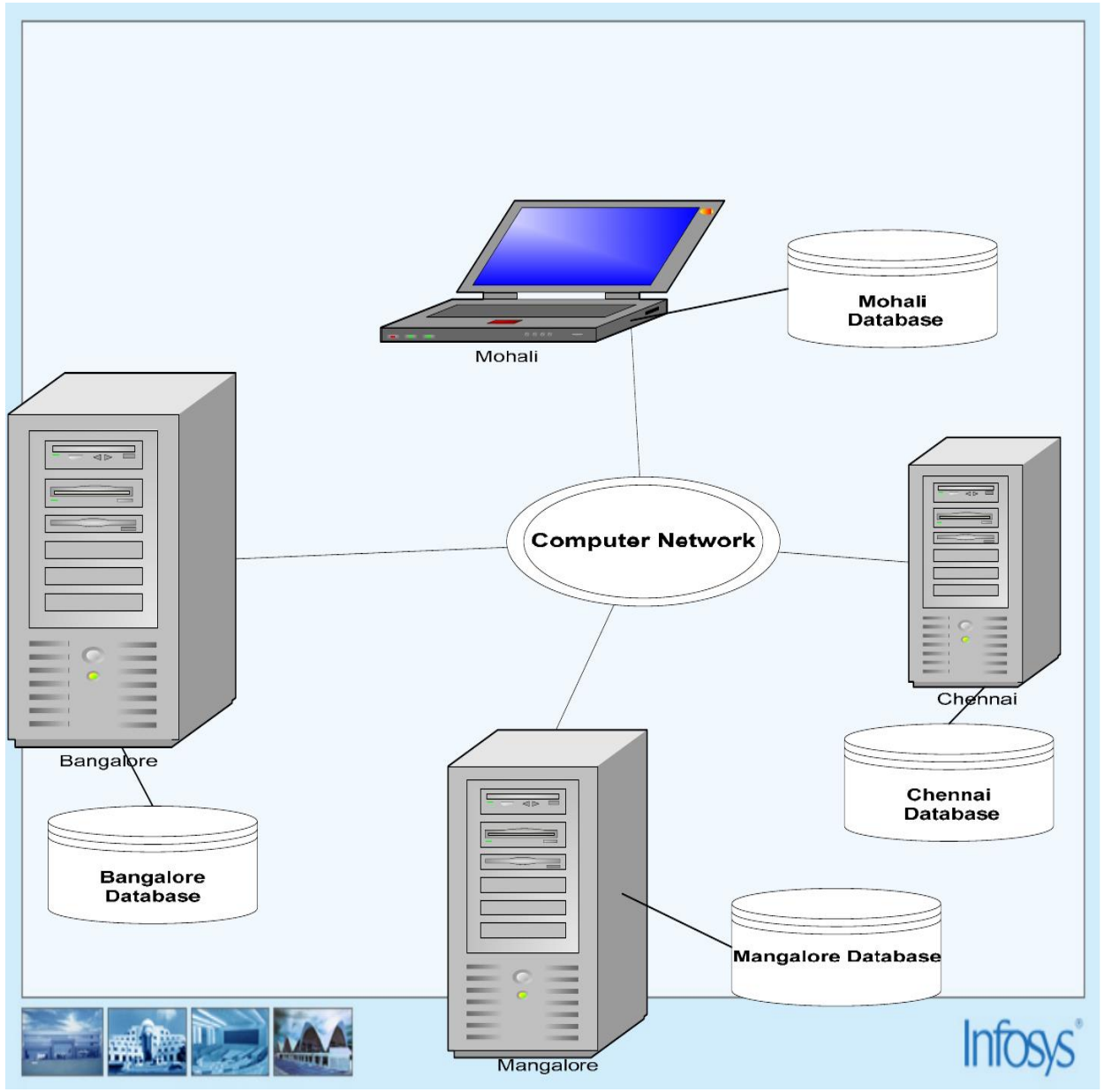
Distributed Database Systems

- Multiple processors (+ memories)
- Heterogeneity and autonomy of “components”
- A distributed database system consists of loosely coupled sites that share no physical component
- Database systems that run on each site are independent of each other
- Transactions may access data at one or more sites

Distributed DBMS Environment

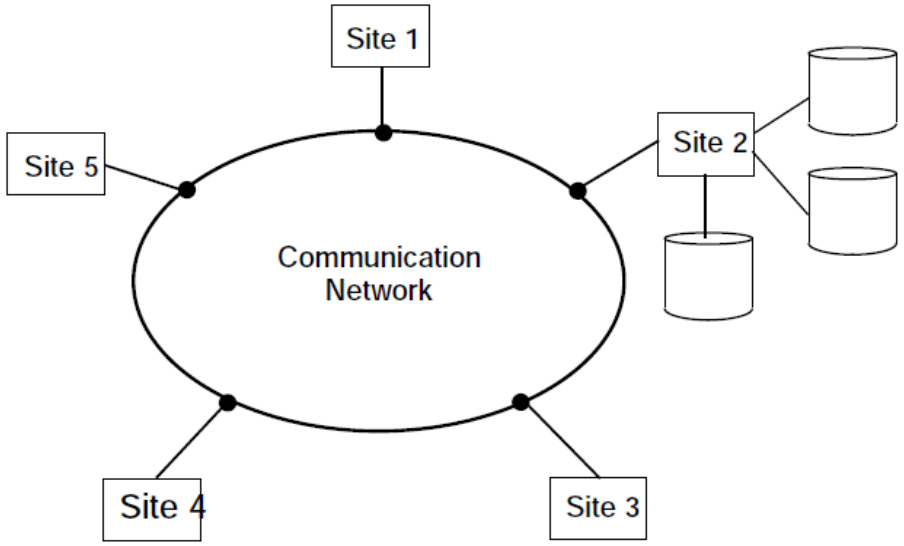



Mohali
Computer Network
Bangalore
Chennai
Mangalore
Corporate Employee Database
Infosys

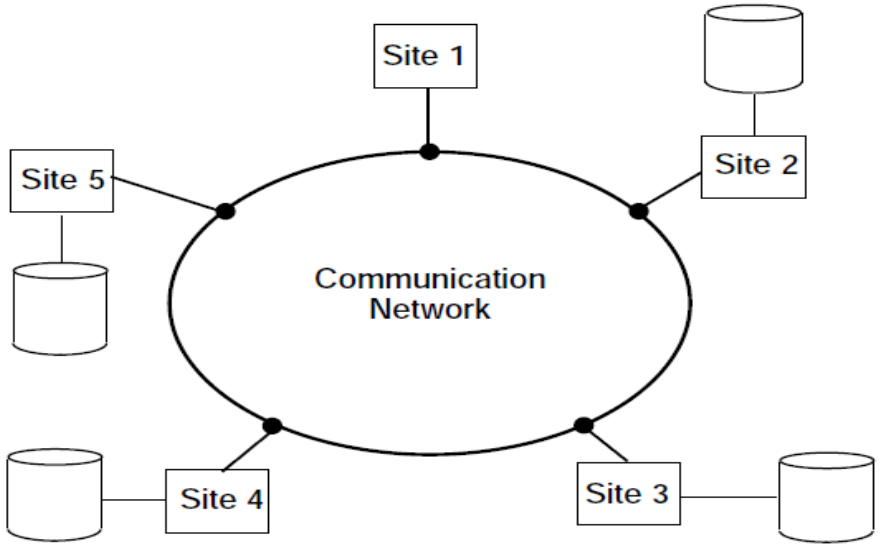

Mohali
Mohali Database
Computer Network
Bangalore
Chennai
Chennai Database
Bangalore Database
Mangalore
Mangalore Database
Infosys

Distributed Data Processing

- Number of autonomous processing elements/logic (may not be homogeneous) interconnected by a computer network cooperates to perform the assigned task.
- Data used by a number of applications may be distributed over number of sites.
- The control of the execution of various tasks might be distributed instead of being performed by one computer system.
- Distributed processing better corresponds to the organizational structure and such a system is more reliable and more responsive. This distribution actually copes up with the large-scale data management problems



Central Database on a Network



DDBS Environment

Characteristics of Distributed Database Management Systems

- Collection of logically-related shared data
- Data split into fragments
- Fragments may be replicated
- Fragments/replicas allocated to sites
- Sites linked by a communications network
- Data at each site is under control of a DBMS
- DBMSs handle local applications autonomously
- Each DBMS participates in at least one global application

(i) **Tightly coupled systems** – In these systems, there is a single systemwide global primary memory (address space) that is shared by all processors connected to the system. If any processor writes some information into the global memory, it can be shared by all other processors in the system. For example, if a processor writes the value 200 to a memory location *y*, any other processor subsequently reading from the location *y* will get the value 200. Thus, in these systems, any communication between the processors usually takes place through the shared memory. The tightly coupled system is illustrated in figure 2.1.

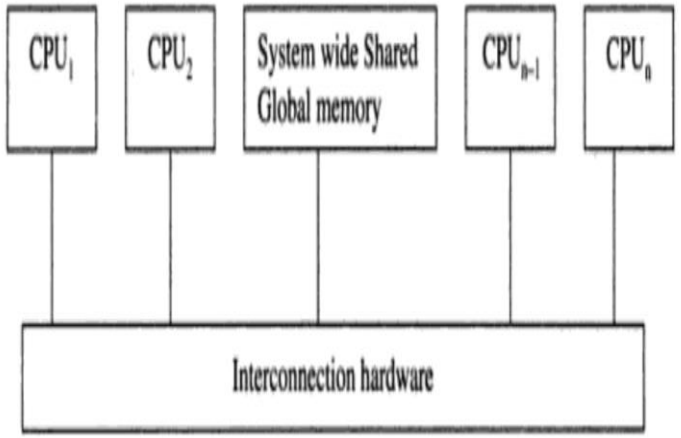


Fig. 2.1 A Tightly Coupled Multiprocessor System

(ii) **Loosely coupled systems** – In these systems, processors do not share memory (clocks and system buses also), and each processor has its own local memory. In this case, if a processor writes the value 200 to a memory location *y*, this write operation only changes the content of its own local memory and does not affect the content of the memory of any other processor. In such systems, all physical communications between the processors are established by passing messages across the network that interconnects the processors of the system. The loosely coupled system is depicted in figure 2.2.

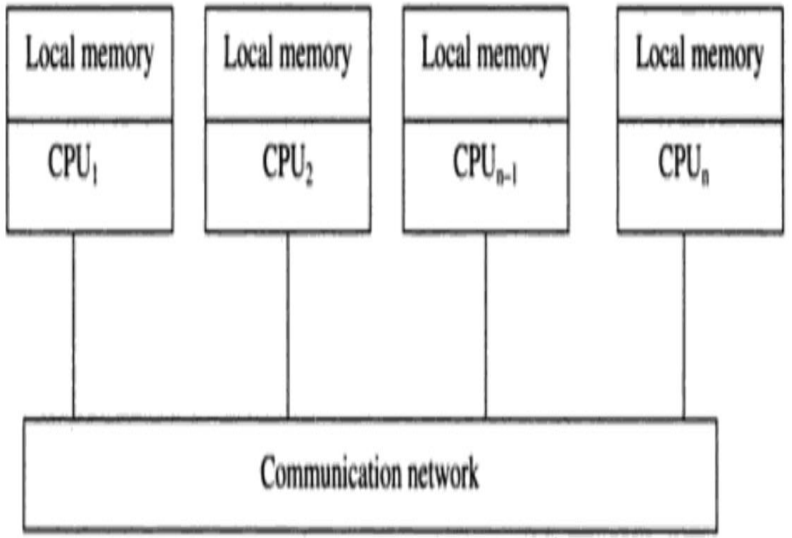


Fig. 2.2 A Loosely Coupled Multiprocessor System

PARALLEL DATABASES

Database systems that run over multiprocessor systems are called as parallel database systems and they are of three main types-

1. **Shared memory (tightly coupled) architecture-**Multiple processors share secondary (disk) storage and also share primary memory.

Distributed DBMS | File-based Data M | What Is a Distribu | Distributed datab | PPT Distributed D | Query Optimizati | Distributed Datab | Distributed Datab
google.co.in/books/edition/Distributed_Database_Systems/0i9uJJ78LtAC?hl=en&gbpv=1&dq=inauthor:"Chhanda+Ray"&printsec=frontcover
Google Books Page 23 Search in this book

when the interconnection network becomes a bottleneck. Shared-memory architecture is illustrated in figure 2.3.

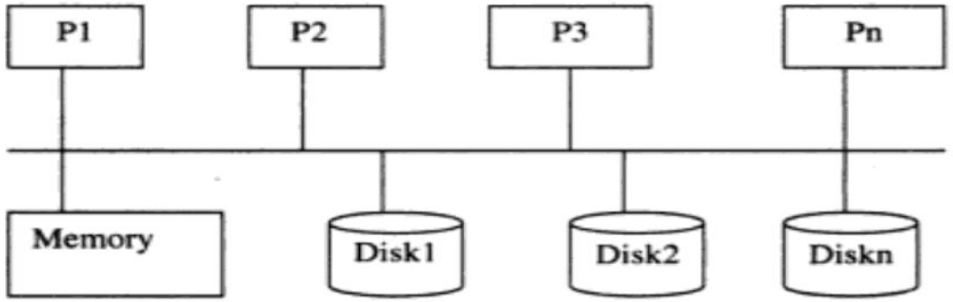


Fig. 2.3 Shared-memory Architecture

Shared-memory architecture provides a number of advantages such as simplicity and load sharing. In this architecture, as all the information is shared by all processors, developing database software is not very difficult, and also it does not vary too much from the database software that is designed for a single-processor system. Load sharing is excellent as it can be achieved at runtime using the shared memory.

30°C AQI 55 ENG 01:04 02-08-2021

PARALLEL DATABASE contd..

2. Shared disk (loosely coupled) architecture- Multiple processors share secondary (disk) storage but each has their own primary memory.

Distributed DBMS | File-based Data M | What Is a Distribu | Distributed databa | PPT Distributed D | Query Optimizatio | Distributed Databa | Distributed Databa
google.co.in/books/edition/Distributed_Database_Systems/0i9uJJ78LtAC?hl=en&gbpv=1&dq=inauthor:"Chhanda+Ray"&printsec=frontcover
Google Books Page 24 Search in this book

in figure 2.4.

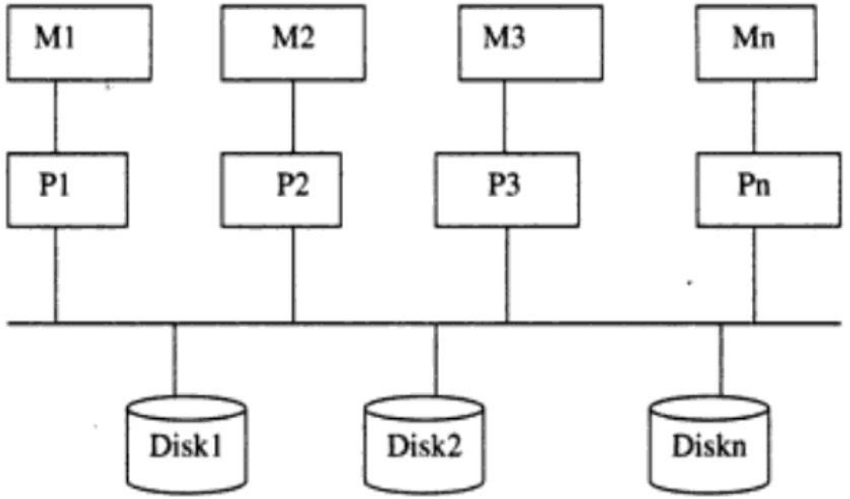


Fig. 2.4 Shared-disk Architecture

DEC (Digital Equipment Corporation) cluster running Rdb was one of the early commercial users of the shared-disk database architecture. Rdb is now owned by Oracle and is called Oracle Rdb.

30°C AQI 55 ENG 01:09 02-08-2021

PARALLEL DATABSE contd..

3. shared nothing architecture- every processor has its own primary and secondary (disk) memory, no common memory exists, and the processors communicate over a high-speed interconnection network (bus or switch)

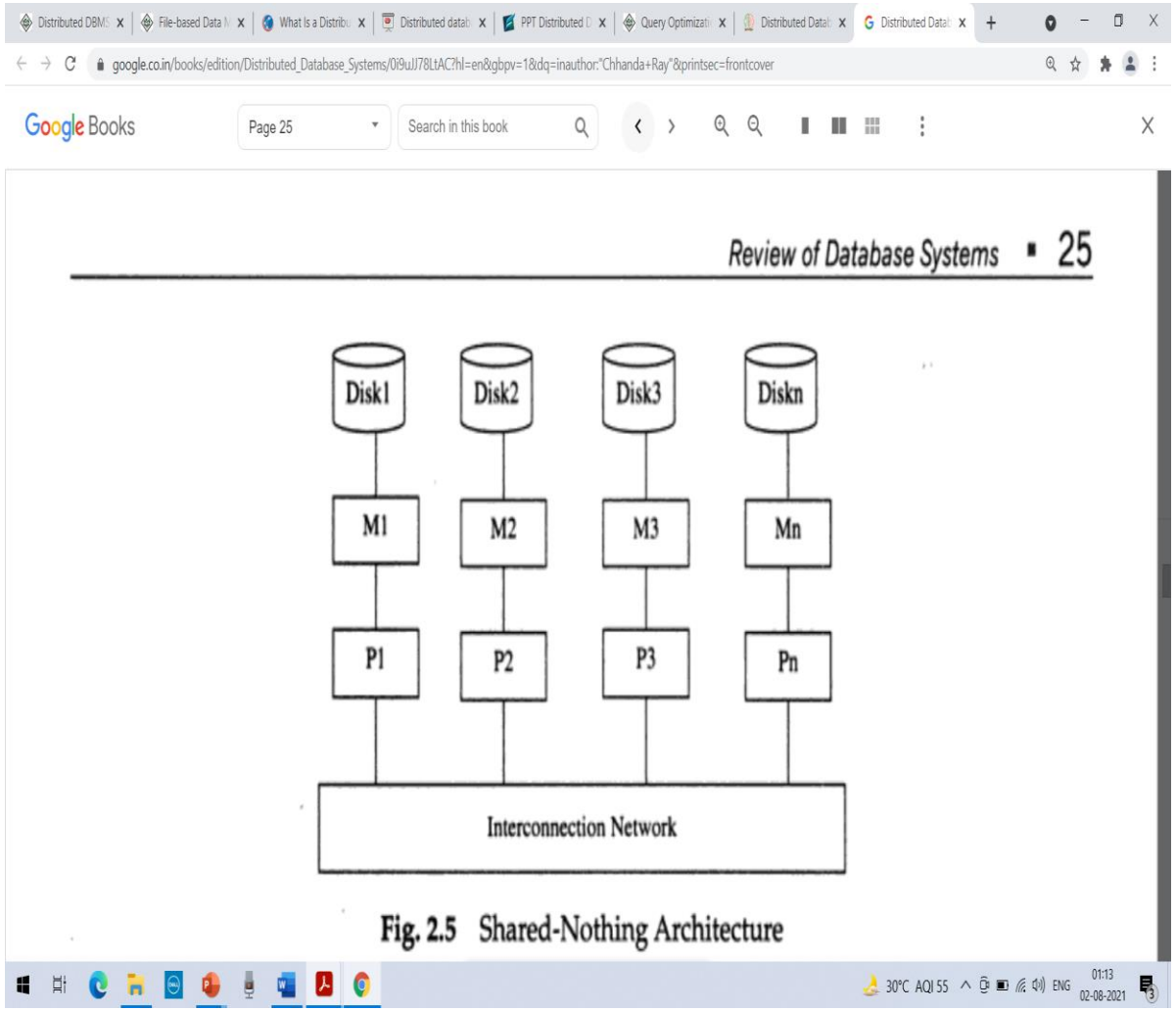
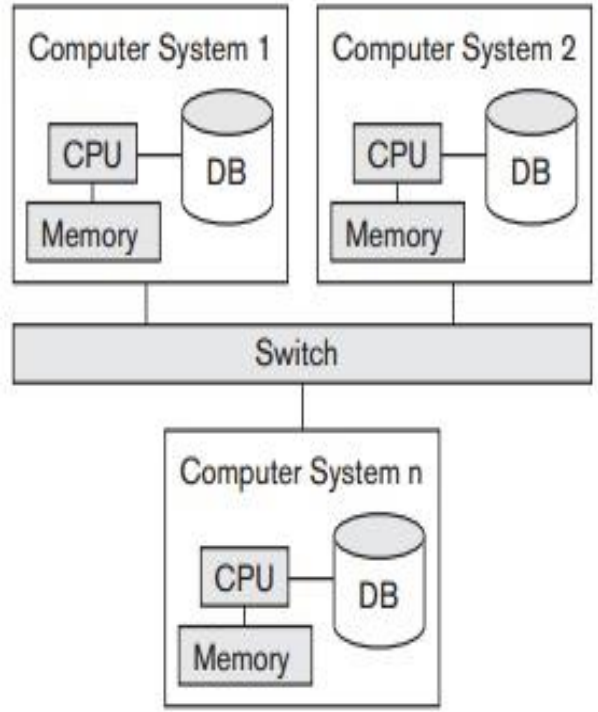


Figure 25.3
Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.

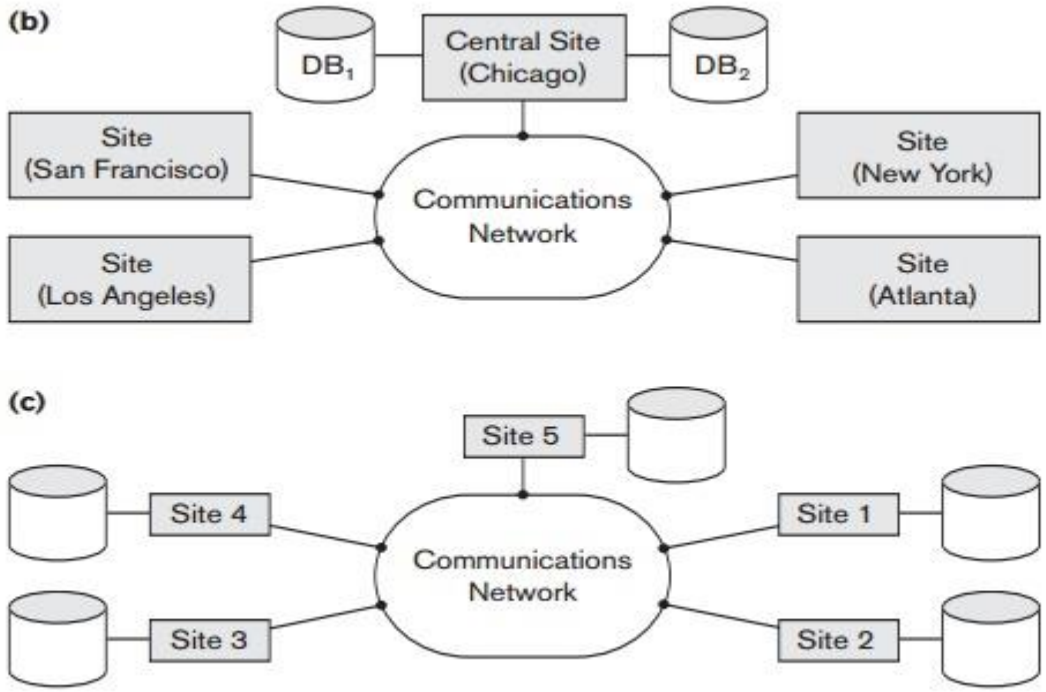


Parallel v/s Distributed Database

- **PARALLEL DATABASE-** A multiprocessor system is symmetrical, consisting of number of identical processors and memory components, and controlled by one or more copies of the same operating system. In shared nothing multiprocessor systems, there is symmetry and homogeneity in the nodes.
- **DISTRIBUTED DATABASE-** For distributed systems, heterogeneity in both operating system as well as hardware is quite common.

What is not a DDBS?

- A timesharing computer system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers - this is a centralized database on a network node



A Distributed DBMS may have a number of local applications, but it has at least one global application. Thus, a distributed DBMS has the following features:

(i) A distributed DBMS is a collection of logically related shared data.

(ii) The data in a distributed DBMS is split into a number of fragments or partitions.

(iii) Fragments may be replicated in a distributed system.

(iv) Fragments/replicas are allocated to different sites.

(v) In a distributed system, the sites are linked by communications network.

(vi) The data at each site is under the control of a DBMS.

(vii) The DBMS at each site has its own right, that is, it can handle local applications independently.

(viii) Each DBMS in a distributed system participates in at least one global application.

Every site in a distributed DBMS may have its own local database depending on the topology of the Distributed DBMS.

29°C Rain showers ENG 02:28 29-07-2021

Distributed Database Features

- **Location independency** - Data is physically stored at multiple sites and managed by an independent DDBMS.
- **Distributed query processing** - Distributed databases answer queries in a distributed environment that manages data at multiple sites.
- **Distributed transaction management** - Provides a consistent distributed database through commit protocols, distributed concurrency control techniques, and distributed recovery mechanisms.
- **Seamless integration** - Collection of interconnected databases residing at different sites basically represents a single logical database.
- **Network linking** - All databases in a collection are linked by a network and communicate with each other.

Functions of Distributed Database

1. **Application Interfaces-** This is used to interact with the end users and remote databases.
2. **Distribution Transparency-** User should be unaware about the distributed nature of the system and a strong trade off between distribution transparency and performance.
3. **Mapping Techniques-** DDBMS must provide mapping techniques to determine the data location of local and remote fragments.
4. **Management of Replicated Data-** DDBMS must have the ability to decide which copy of a replicated data item to be selected while executing a data request.
5. **Extended Query Processing & Optimization-** DDBMS provides query optimization for both local & global queries to find the best access strategy.

Functions of Distributed Database contd ..

6. Distributed Transaction Management- Consistency of data should not be violated by local and distributed transactions.

7. Distributed Backup & Recovery Services- These services ensures availability and reliability of a database in case of failures.

8. Support for Global System Catalog- DDBMS must contain a global system catalog to store data distribution details for the system.

9. Support for Global Database Administrator- This is responsible for maintaining the overall control of data and programms in DDBMS.

10. Distributed Security Control- This feature is used to maintain appropriate authorization/access privileges to the distributed data.

DATE'S 12 OBJECTIVE RULES FOR DDBMS

1. **Local Autonomy-** Operations at a particular site should be managed by that particular site and no site should depend on some other site.
2. **Non-Reliance on a Central Cite-** No site relies on a central cite, i.e there should be no one site without which the system cannot operate.
3. **Continuous operations –** Ideally, there should never be a requirement for a planned system shutdown due to various database related operations.
4. **Location Independence-** users are unaware about the physical storage of the data and they can access the data from all sites.
5. **Fragmentation Independence-** Users are unaware of data fragmentation and will be able to access all data no matter how it is fragmented.
6. **Replication Independence-** Users are unaware that the data item is replicated and neither they can access a particular data item copy nor they can modify all data copies.

DATE'S 12 OBJECTIVE RULES FOR DDBMS contd..

7. Distributed Query Processing- Distributed queries can reference data form more than one site and query optimization is performed transparently.

8. Distributed Transaction Processing- both local and global transactions must ensure ACID properties.

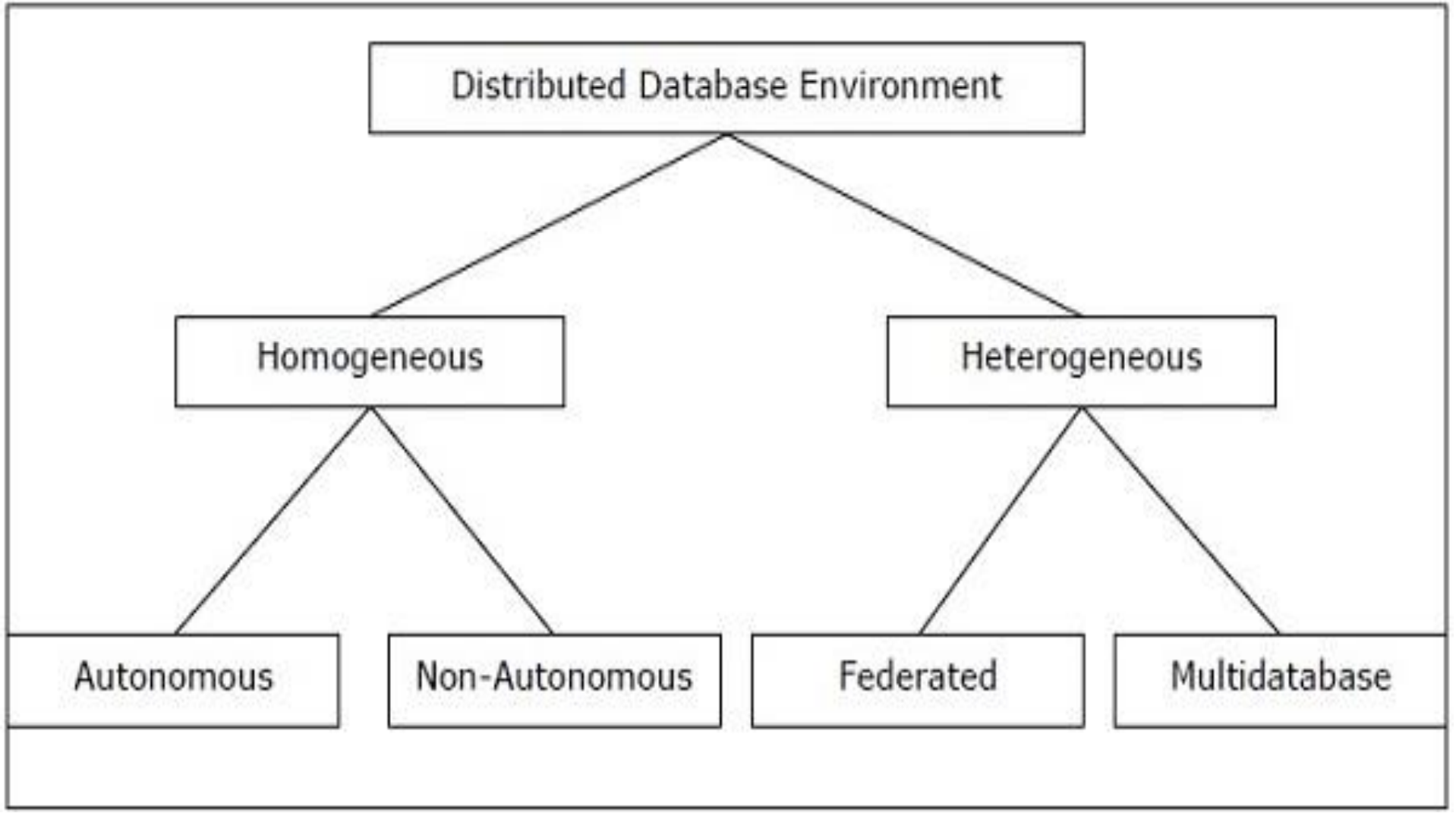
9. Hardware Independence- DDBMS should run on a Varity of H/w platforms.

10. Operating System Independence- DDBMS should run on a Varity of operating systems.

11. Network Independence- It is possible the DDBMS run on a Varity of communication networks.

12. Database Impendence- DDBMS can be made up of different local DBMS set ups, i.e supporting different underlying data models.

Types of Distributed Databases



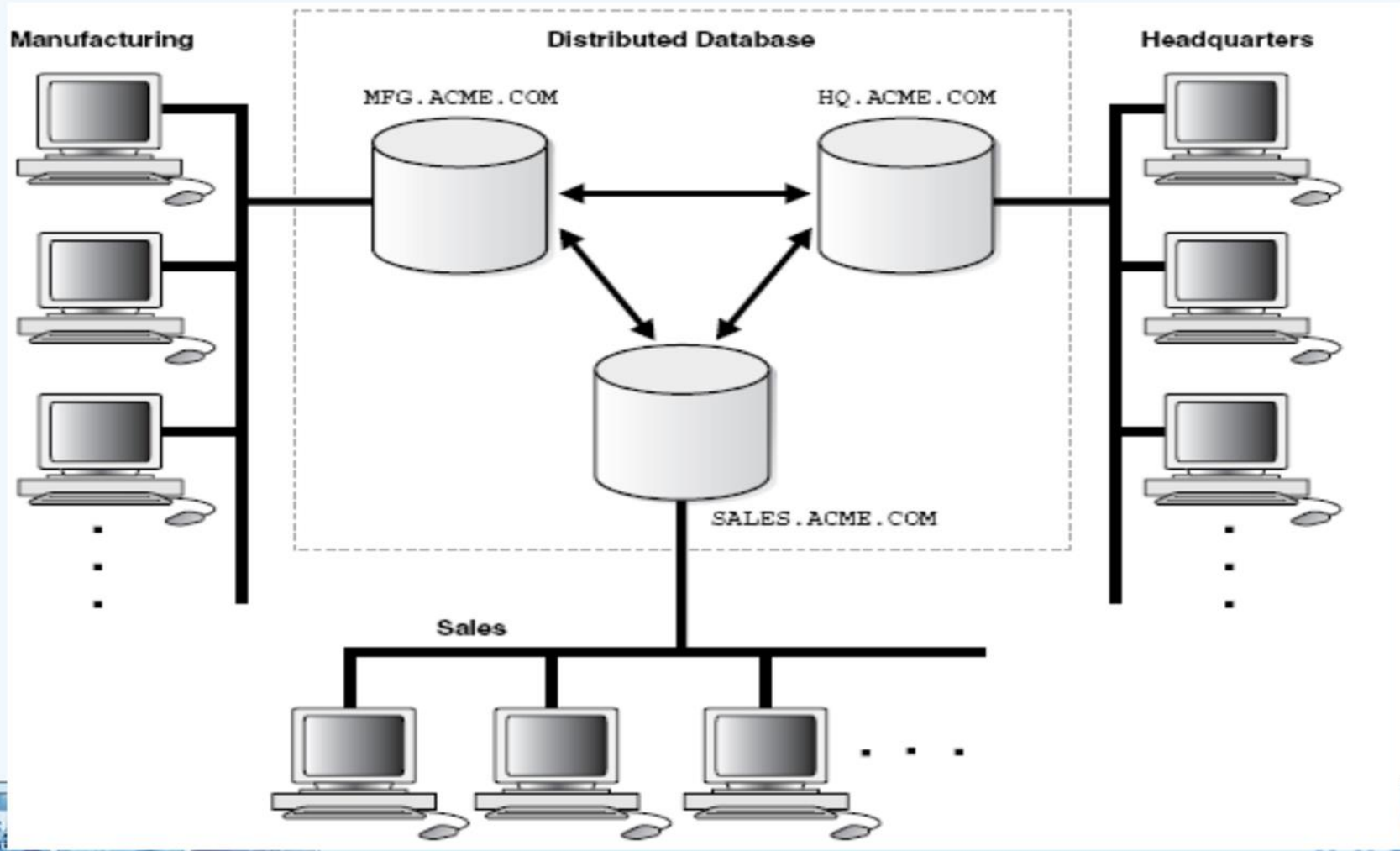
HOMGENEOUS DISTRIBUTED DATABASE

- In an ideal scenario all sites will share a common global schema, (although some relations may be stored only at some local sites) and run the same DBMS software.
- Sites are aware each other sites and they cooperates in processing the user requests.
- Mostly same identical operating systems even if the processors are not same.
- Each site surrenders autonomy in terms of right to change the schema or software
- Appears to the user as a single system and the database is accessed through a single interface.
- Homogeneous DDBMS provides several advantages such as-

- Simplicity,
- ease of designing and management
- incremental growth.

These systems improve performance by exploiting parallel processing capabilities of multiple homogenous nodes.

HOMGENEOUS DISTRIBUTED DATABASE

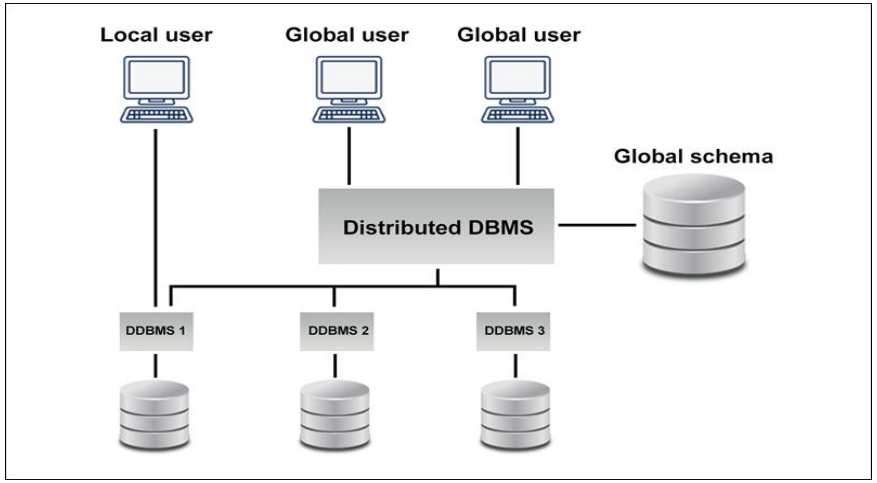


Types of Homogeneous Distributed Database

- 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.

HETEROGENEOUS DISTRIBUTED DATABASE

- Sites may run different operating systems and DBMS that need not be following same underlying data model (i.e relational, hierarchical, object oriented).
- Sites having their own local databases and hence different schemas and products.
- Each site is completely unaware of other sites and hence limited cooperation among those sites in processing user requests.
- To allow communication among the different sites interoperability between different DBMS products i.e translations are required.



- Heterogeneity may occur at different levels-

1. **Different Hardware-** If the hardware is different but the DBMS software is same then translation is straightforward.
2. **Different DBMS Software-** Here the execution of global trnsactions are very complicated as it involves mapping of one data structures in one model to the equivalent data structures in another model. The translation of query language is also essential.
3. **Different Hardware and different software-** It is extremely complex as translation of both hardware and DBMS software are required.

The provision of a common conceptual schema which is formed by integrating individual local conceptual schemas, adds extra complexity to the distributed processing.

To address such heterogeneity, gateways are used in some relational systems which basically works as a query translator, but it may not support transaction management and homogenizing the structural and representational differences between different DBMSs.

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.

FEDERATED DATABASE SYSTEM (FDS)

- Several decentralized autonomous databases appear to function as a single entity and FDS transparently integrates multiple autonomous database systems into a single federated database.
- A federated database, or virtual database, is the fully integrated, logical composite of all constituent databases in a federated database system
- FDS must be able to decompose the query into subqueries for submission to the relevant constituent DBMS's, after which the system must composite the result sets of the subqueries.
- Because various DBMS employ different query languages, so FDS can apply wrappers to the subqueries to translate them into the appropriate query languages
- The goal of the system is to ensure that a typical query will need to use only one component, thus drastically reducing the number of rows that need to be searched

DDBMS DESIGN ISSUES

- These are applications where data and its accesses are inherently distributed in different sites to increase the availability during failures.
- The methodology for designing DDBMS is same as that of centralized database. However, some additional factors are considered here such as-

A. Data Replication- If relation r is replicated, a copy of relation r is stored in two or more sites. This concept having a number of advantages and disadvantages

(i) Availability- If one of the site containing relation r fails, then relation r can be found in other sites and the system can continue to process queries involving r.

(ii) Higher parallelism- For read operation on relation r, several sites can process queries involving r in parallel and hence less data movement between sites.

(iii) Increased overhead on update- The system must ensure that all replicas are consistent. Thus, whenever r is updated, the update must be propagated to all the sites containing its replicas of r.

DDBMS DESIGN ISSUES (contd..)

B. Data Fragmentation- Database may be broken up into logical units called fragments which will be stored at different sites. Three Types of Data Fragmentations are:

- **Horizontal fragmentation:** It divides a table 'horizontally' by selecting the relevant rows and these fragments can be assigned to different sites.
- **Vertical fragmentation:** A vertical fragment of a table keeps only certain attributes of it. It divides a table vertically by columns and these fragments are stored in different sites. It is necessary to include the primary key of the table in each vertical fragment so that the full table can be constructed if needed.
- **Hybrid fragmentation:** It comprises the combination of both Horizontal and Vertical Fragmentation. Each fragment can be specified by a SELECT-PROJECT combination of operations. In this case the original table can be reconstructed be applying union and natural join operations in the appropriate order.

DDBMS DESIGN ISSUES (contd..)

C. Data Transparency

The user of a distributed database system should not be required to know either where the data are physically located or how the data can be accessed at the specific local site. This is called data transparency and it can take several forms:

Fragmentation transparency- Users are unaware about how a relation is fragmented.

Replication transparency- Users do not have to be concerned with what data objects have been replicated, or where the replicas have been placed.

Location transparency- Users are not required to know the physical location of the data. The distributed database system should be able to find any data as long as the data identifier is supplied by the user transaction.

FDS ARCHITECTURE

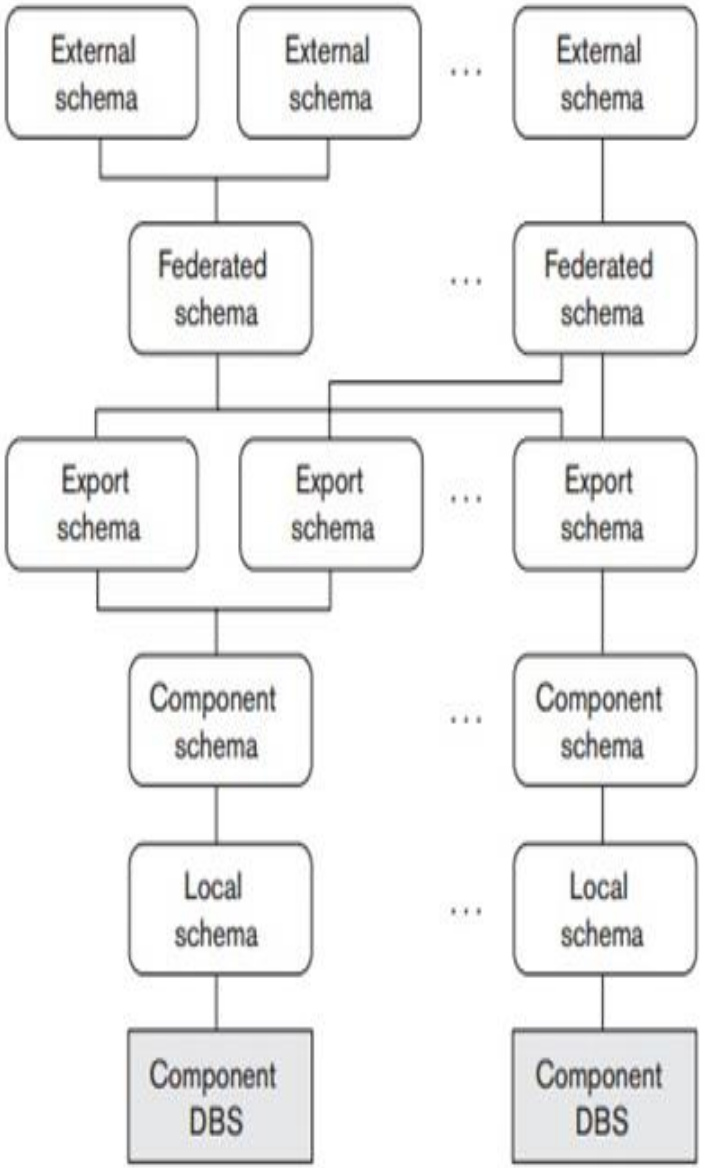


Figure 25.6
The five-level schema architecture in a federated database system (FDBS).

Source: Adapted from Sheth and Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases." *ACM Computing Surveys* (Vol. 22: No. 3, September 1990).

The five level schema architecture-

- Local Schema is the conceptual concept expressed in primary data model of component DBMS.
- Component Schema is derived by translating local schema into a model called the canonical/common data model.
- Export Schema is a subset of component schema (contain access control information) and it helps in managing the flow of control of data.
- Federated Schema is an integration of multiple export schema and it includes information on data distribution generated when integrating export schemas.
- External Schema defines a schema for a user/applications or a class of users/applications.

ADVANTAGES OF DISTRIBUTED DATABASE

1. **Sharing of Information-** User at one site accesses data residing at other sites.
2. **Faster Data Accessing-** If end user requested data available locally, then faster data accessing as compare to the remotely located centralized system. Also it is possible to split a query into a number of sub queries that can be executed in parallel at different sites.
3. **Increased Availability & Reliability-** If a transaction accessing data item form a failed site, then it can find it form other sites due to data replication and hence non termination of the transaction.
4. **Processor Independence-** Since users can access any available copy of the data item, so user request do not depend on a specific processor.
5. **Modular Extensivity-** new sites can be added to the network without affecting the operations of other sites. Such flexibility allows organizations to extend their system in a relatively rapid and easier way.

PROMISES OF DDBMS

1. Transparency Management for Distributed and Replicated Data

- Transparent system "hides" the implementation details from the end user and this concept helps in building the complex applications.
- Storing each partition of a relation at different sites is known as fragmentation.
- Furthermore, it may be preferable to duplicate some of these fragmented data items at different sites for performance and reliability reasons.
- Fully transparent access means that the user can still pose the query, without paying any attention to the fragmentation, location or replication of the data.

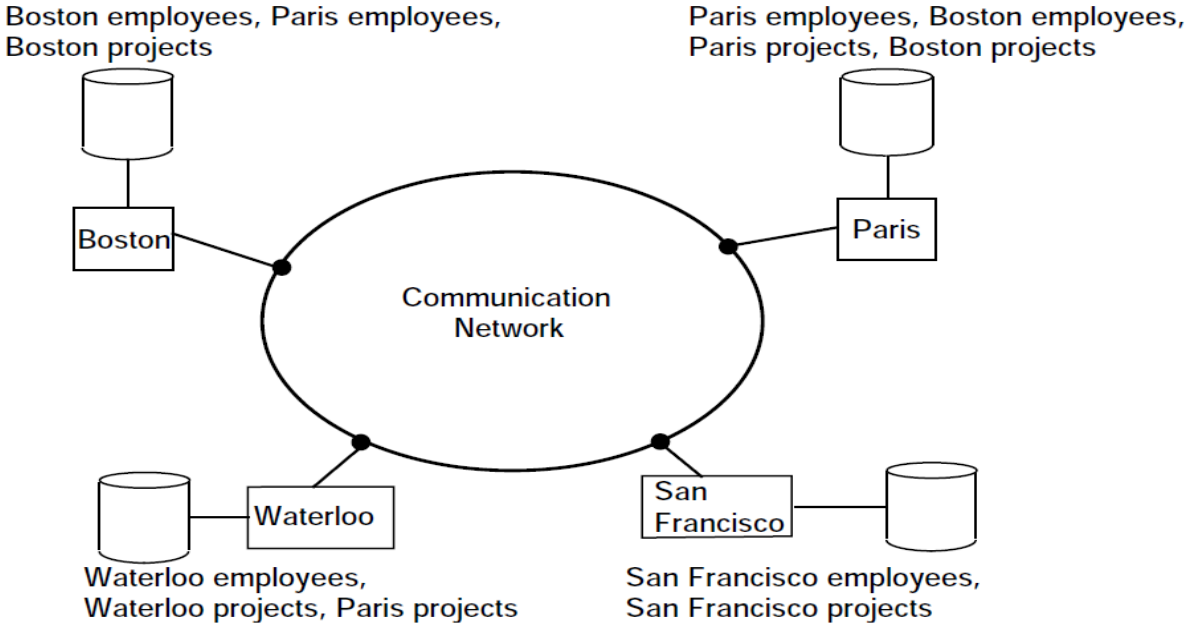


Fig. 1.5 A Distributed Application

1. **EMP**(ENO, ENAME, TITLE)
2. **PROJ**(PNO, PNAME, BUDGET)
3. **SAL**(TITLE, AMT)
4. **ASG**(ENO, PNO, RESP, DUR)

Query:

SELECT ENAME, AMT
FROM EMP, ASG, SAL
WHERE ASG.DUR > 12
AND **EMP.ENO = ASG.ENO**
AND **SAL.TITLE = EMP.TITLE**

Different Forms of Transparencies

Data Independence- implies immunity to the user applications such that any changes in data definition and organization will be having no impact to the end user level.

Logical data independence –refers immunity to the user applications related to changes in the logical structure (i.e., schema) of the database.

Physical data independence- deals with hiding the details of the storage structure from user applications.

User applications should not be concerned with the details of physical data organization. Therefore, the user application should not need to be modified when the data organization changes.

Network Transparency

User should be protected from operational details of the network.

There will be no difference between database applications run on centralized database and distributed database.

Network transparency categorization from the viewpoint of-

1. **Network services provided-** uniform means by which services are accessed.
2. **Distribution Transparency (DBMS perspectives) -** requires that users do not have to specify where the data is located.

Sometimes two types of distribution transparencies are identified:

A. **Location transparency-** command used to perform the task is independent of both the location of the data and the system on which the operation is being carried out.

B. **Naming transparency-** unique name is provided for each object in the database. In the absence of naming transparency, users are required to embed the location name (or an identifier) as part of the object name.

Replication Transparency

- For performance, reliability, and availability, it is usually desirable to distribute the data in a replicated fashion across the network.
- Replication transparency refers only the existence of replicas, not to their actual locations.
- Distributing these replicas across the network in a transparent manner is the domain of network transparency.

Fragmentation Transparency

- Relations are divided into smaller fragments and each fragment is treated as a separate object (i.e., another relation) for performance, availability and reliability.
- In fragmentation we have to deal with the problem of handling user queries that are specified on entire relation. Typically, this requires a translation from a global query to several fragmental queries.
- Fragmentation transparency is mainly related to query processing.

DIFFERENT LAYERS OF TRANSPARENCY

The level of transparency is inevitably a compromise between the ease of use and the difficulty and overhead cost of providing high levels of transparency.

Language Transparency (API level)– requested service in the user language is converted to the required operations which is to be taken care of by the compiler or interpreter. This layer provides high level data access to the user. Here no transparent service is provided to the implementer of compiler or interpreter.

- **In the second layer** management of network resources is taken over by the distributed operating system or the middleware.
- **The third layer** at which transparency can be supported is within the DBMS. It is the responsibility of the DBMS to make all the necessary translations from the operating system to the higher-level user interface.

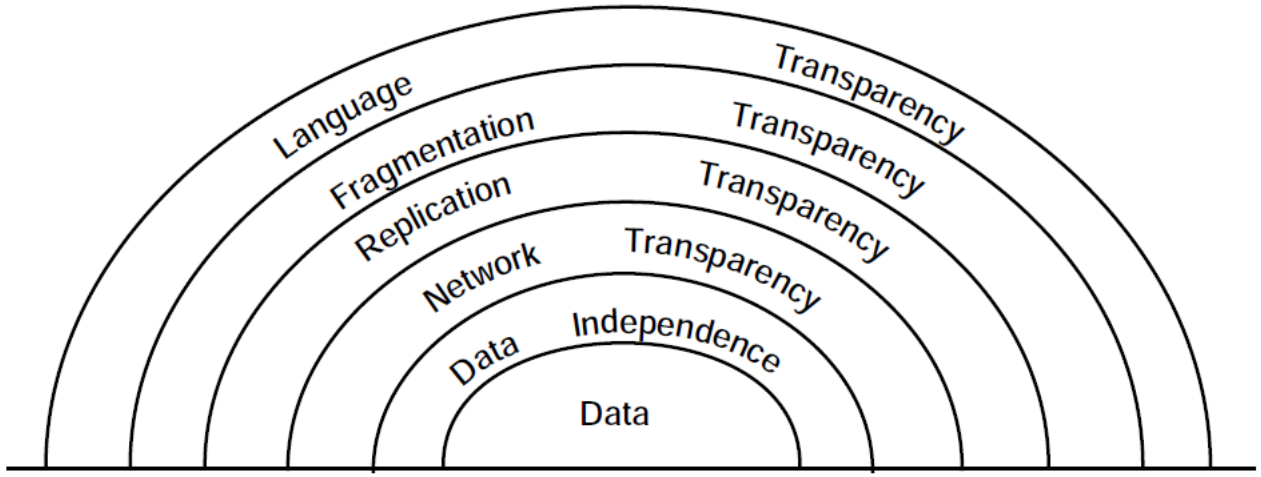


Fig. 1.6 Layers of Transparency

2. RELIABILITY THROUGH DISTRIBUTED TRNSACTIONS

- Distributed DBMSs are intended to improve reliability since they have replicated components and, thereby eliminating the single point of failure.
- Distributed transactions are executed at a number of sites which actually accesses the local database.
- user applications do not need to be concerned with coordinating their accesses to individual local databases nor do they need to worry about the possibility of site or communication link failures during the execution of their transactions.

3. Improved Performance

Improved performance for distributed DBMSs is based on two points.

1. Data localization- Data to be stored in close proximity to its points of use. This has two potential advantages:

(i) Since each site handles only a portion of the database, so contention for CPU and I/O services is not as severe as for centralized databases.

(ii) Localization reduces remote access delays that are usually involved in wide area networks

2. Inherent parallelism of distributed systems-

(i) Inter-query parallelism- ability to execute multiple queries at the same time.

(ii) intra-query parallelism- breaking up a single query into a number of subqueries each of which is executed at a different site, accessing a different part of the distributed database.

4. Easier System Expansion

- In a distributed environment, it is much easier to accommodate increasing database sizes.
- expansion can usually be handled by adding processing and storage power to the network.
- One aspect of easier system expansion is economics. It normally costs much less to put together a system of “smaller” computers with the equivalent power of a single big machine.

COMPLICATIONS ASSOCIATED WITH DDBMS

1. **Need for Complex and expensive software:** DDBMS often demands complex and more expensive software for data transparency and also co-ordination across several sites.
2. **Processing overhead:** Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
3. **Data integrity:** The need for updating the duplicated data in multiple sites mainly poses problems. In this context, DDBMS is responsible for-

   (i) choosing one of the stored copies of the data for access in case of retrievals

   (ii) making sure that the update is reflected on every copy of the data item.

4. Dealing with Site Failure: If some site fails or some communication link fails while an update is going on, the system must make sure that the updated effects will be reflected on the concerned data item residing at the failing site as soon as the system recovers from the failure.

COMPLICATIONS ASSOCIATED WITH DDBMS (contd..)

5. Maintaining Synchronization Among Sites: Since each site cannot have instantaneous information on the actions currently being carried out at the other sites, so maintaining synchronization among different sites are considerably harder than the centralized system.

6. Security concerns: Since data exchange occurs among different sites so maintaining proper security of the exchanged data is a big challenge apart from secure communication channel problem.

7. Overheads associated with improper data distribution: Responsiveness of queries largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

Distributed DBMS Architecture

Architecture of a system deals with

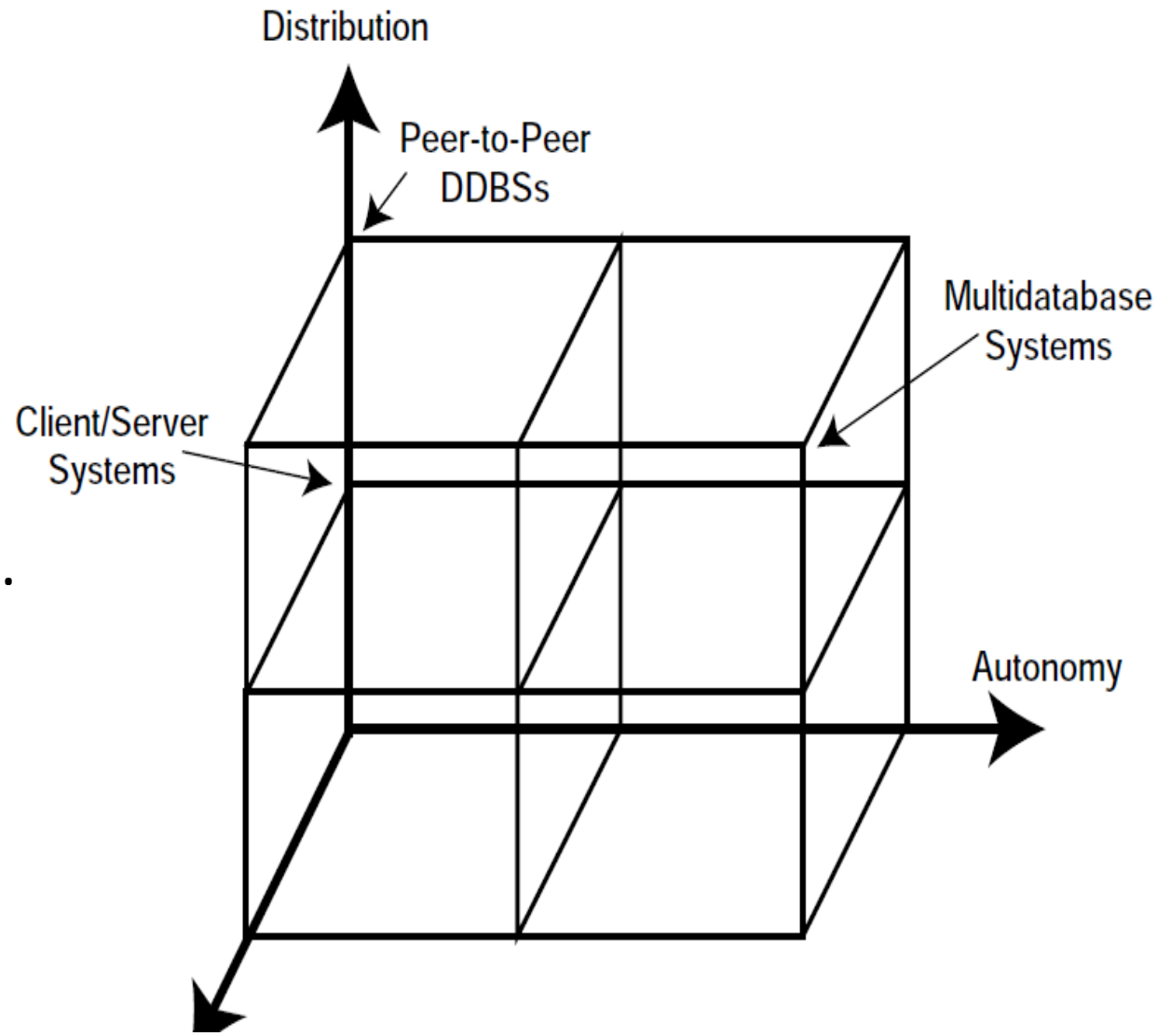
1. Components of the system
2. Functions of the system
3. Interaction between all components

DDBMS architectures are generally developed depending upon three parameters:

1. Distribution: It states the physical distribution of data across the different sites.

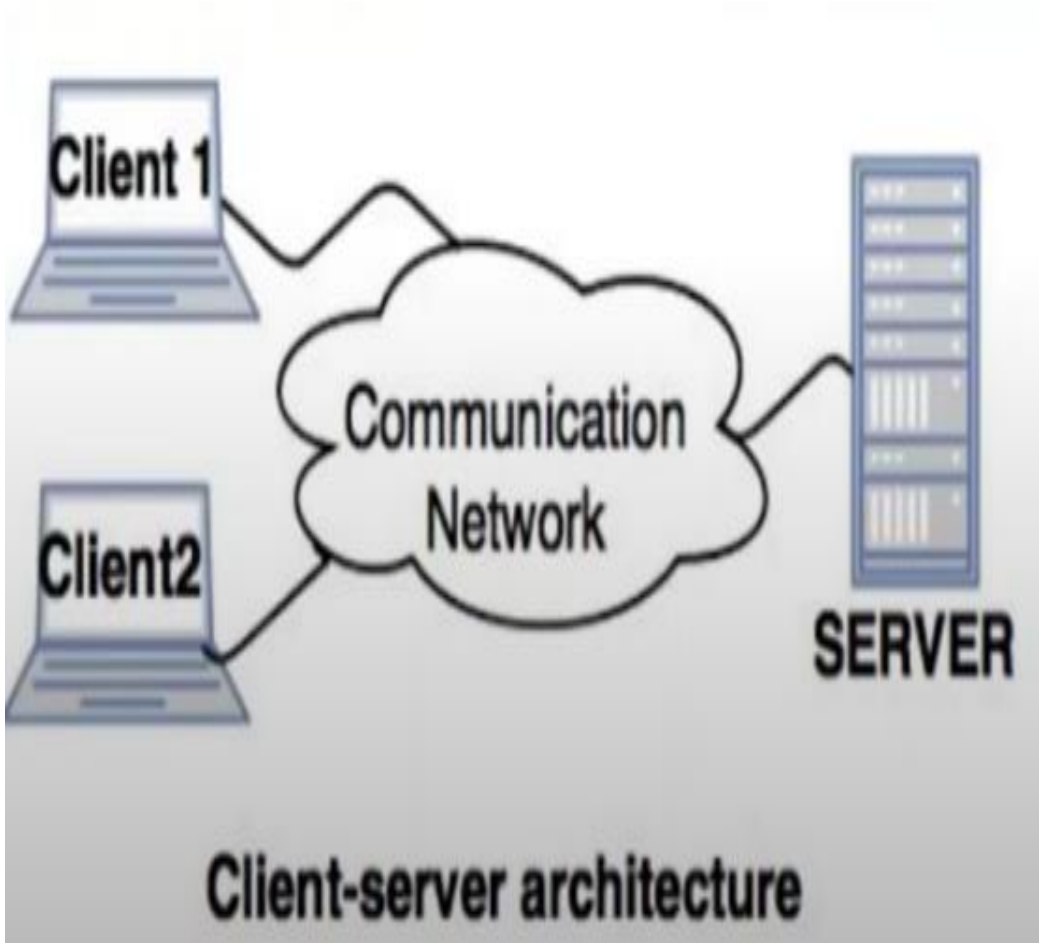
2. Autonomy: It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.

3. Heterogeneity: It refers to the uniformity or dissimilarity of the data models, system components and databases.



Client Server Architecture

- It has a number of clients and a few servers connected via a network
- A client sends a query to one of the servers. The earliest available server solves it and replies.
- It is simple to implement and execute due to centralized server systems.



Client-Server Architecture with Client and Server Roles

- This is a two-level architecture where the functionality is mainly divided into servers and clients. This two level concept basically promotes easier management of complex DBMS systems and also the complexity of data distributions.
- The primary server functions are mainly ---

data management, query processing,

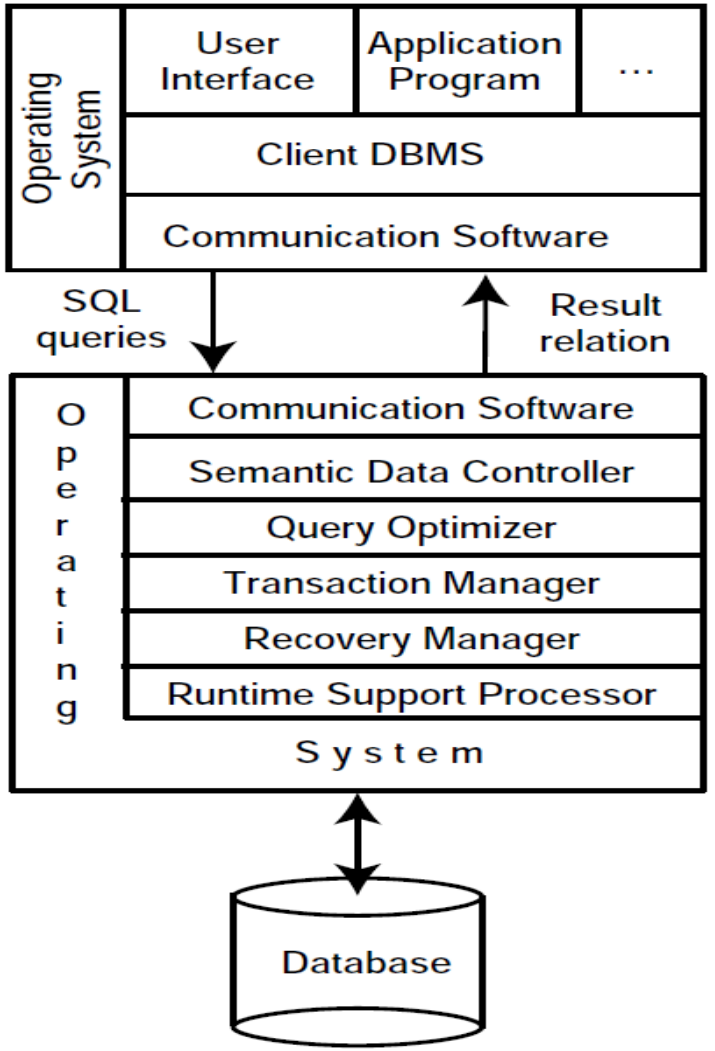
query optimization and transaction management.

- Client functions mainly include user interface. However,

they also performs consistency checking of user queries

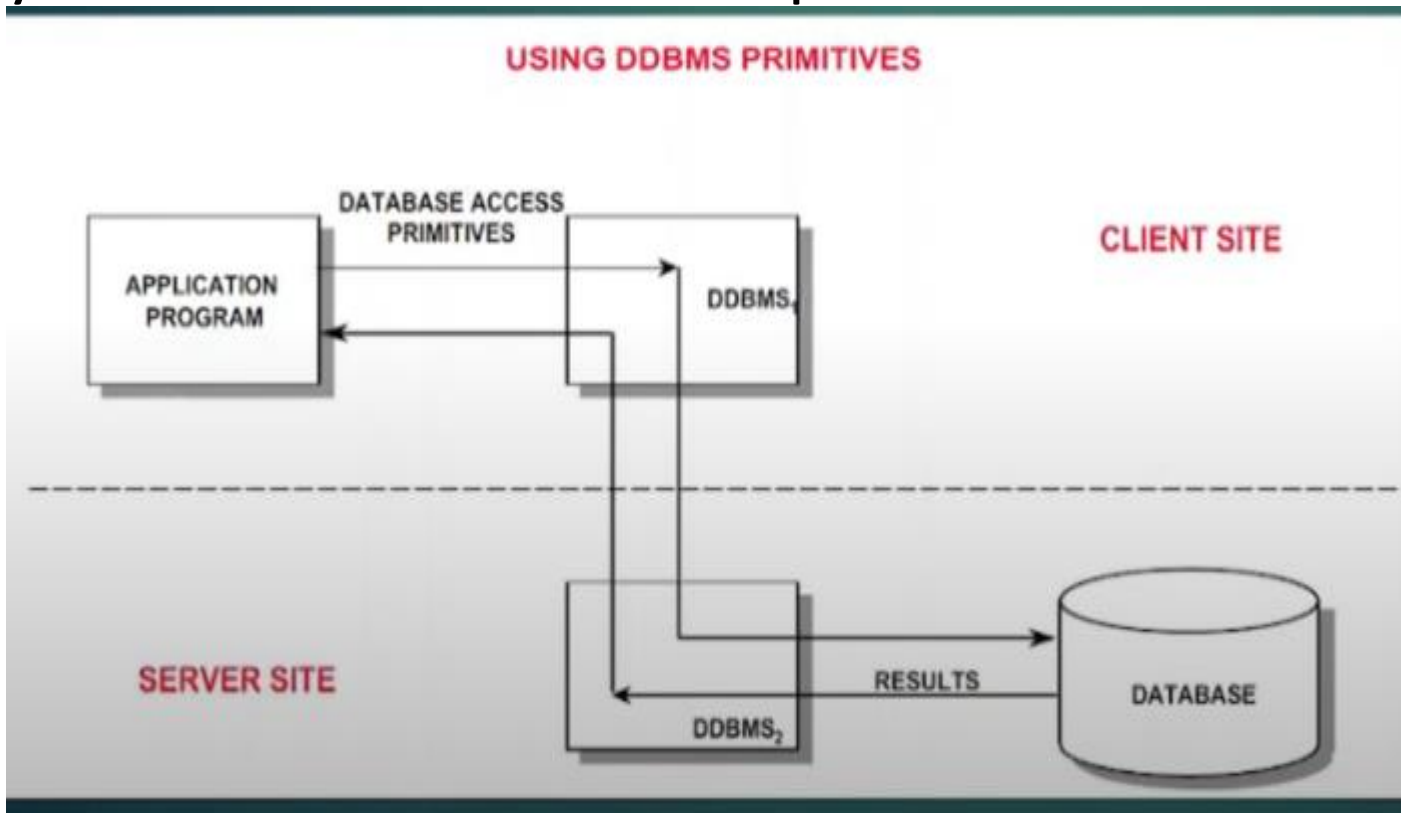
and at times the transaction management.

- The two different types of client - server architectures
- Single Server-Multiple Client (centralized systems)
- Multiple Server-Multiple Client



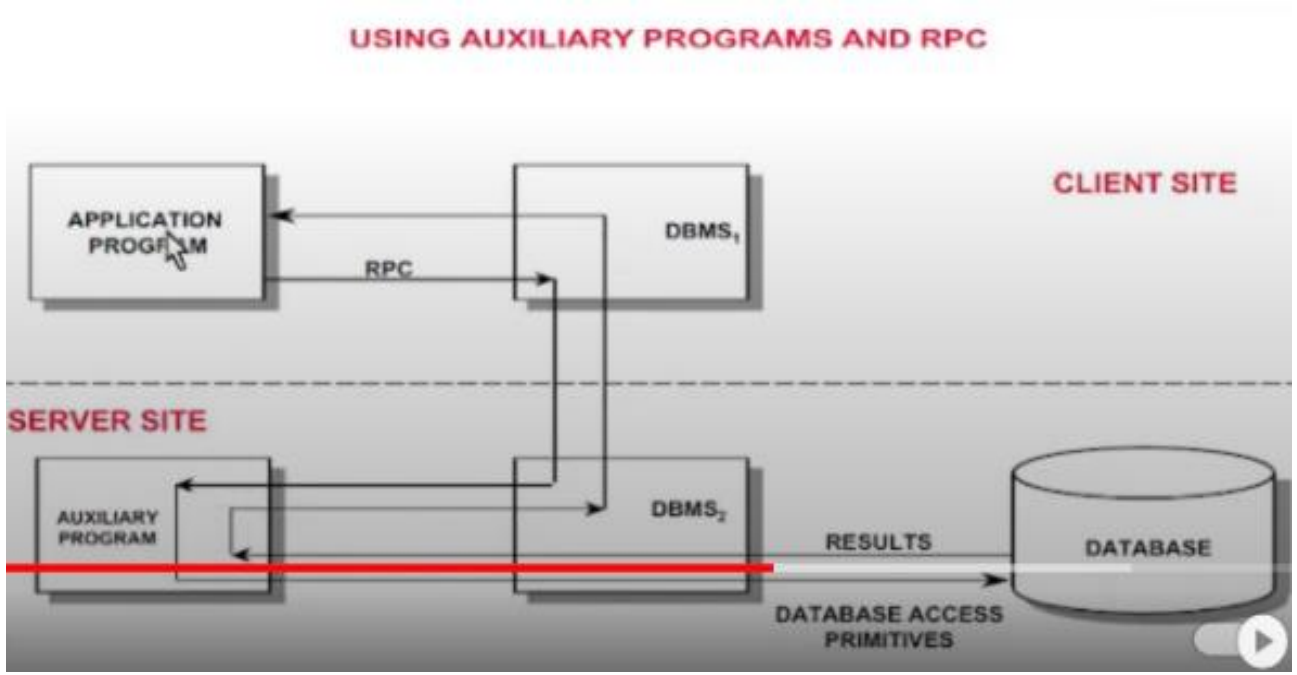
Types of Client server Architecture

- **Using DDBMS primitives** – client local DDBMS component routes the query to to server which accesses the DB and sends back the results.
- High distribution transparency (as file names are global) and works like a centralized system
- Low efficiency as answer travels one tuple at a time.



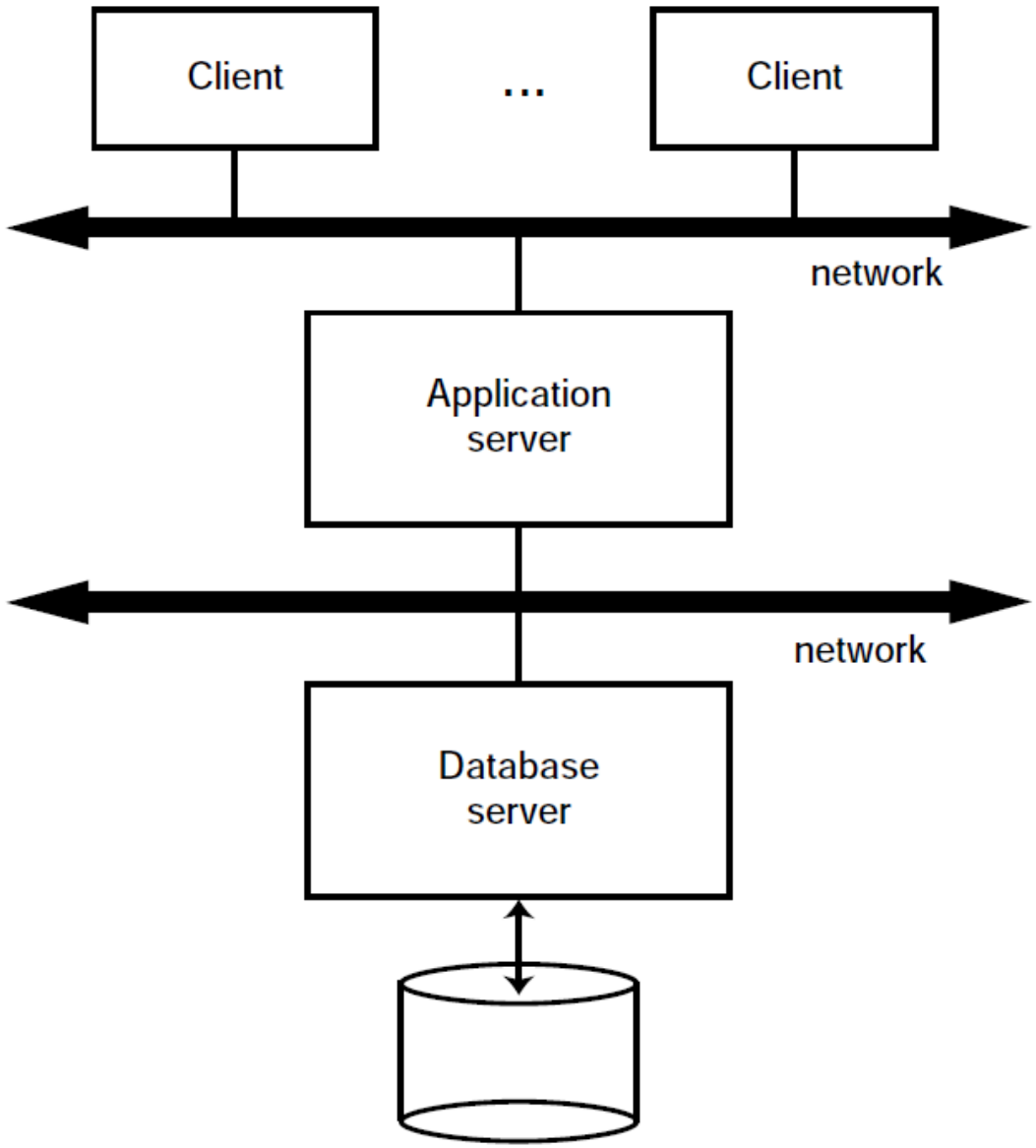
Types of Client server Architecture contd..

- **Using auxiliary programs & RPC** –Remote database can also be accessed by an via an auxiliary program. This prog. is written by an application programmer which returns the results to the requesting application.
- The applications asks the auxiliary program to excute on the server and to send back the result.
- The auxiliary prog. assembles tuples into result sets improving transmission efficiency.



Client-Server Architecture contd ..

- Single server multiple client scenario



Client-Server Architecture contd

Multiple Server-Multiple Client scenario- two alternative management strategies-

1. each client manages its own connection to the appropriate server

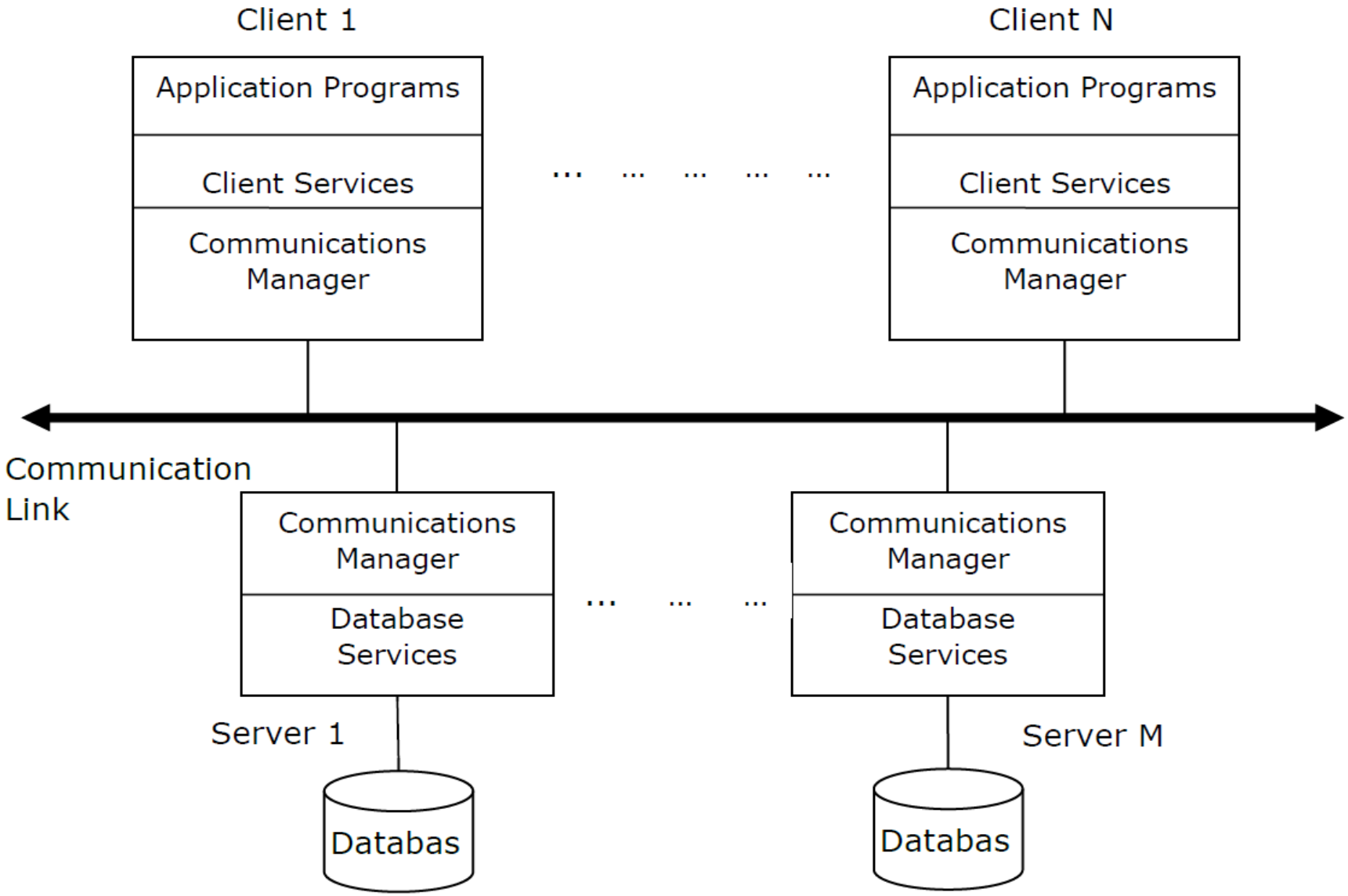
   OR

2. each client knows only its “home server” which then communicates with other servers as required.

- The first approach loads the client machine with additional responsibilities and this is known as “heavy client” system.
- The second approach leaves the data management functionality for the servers. Thus, transparency of data access is promoted and this is mainly “light clients.”

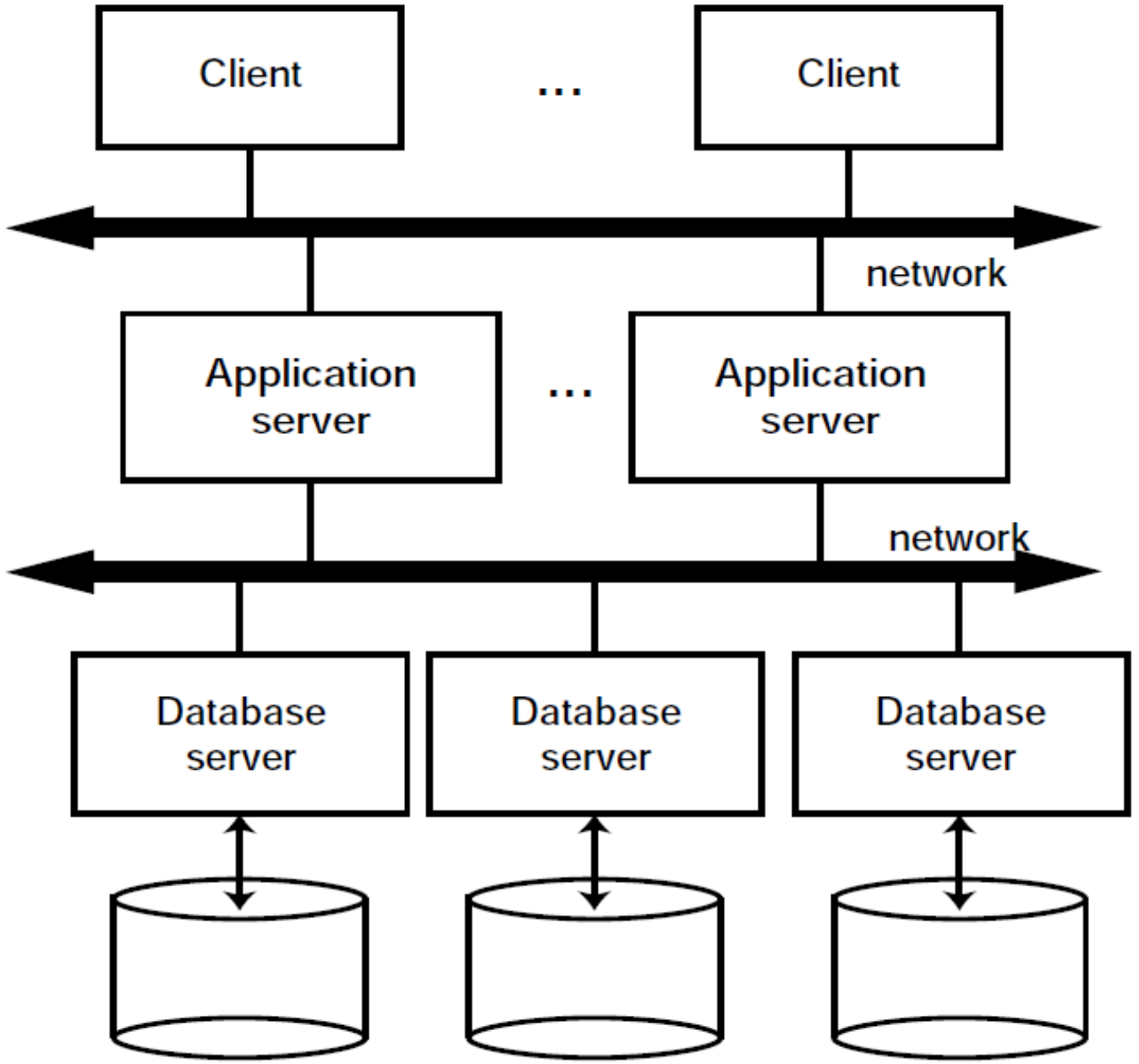
Client-Server Architecture contd..

- Heavy client systems



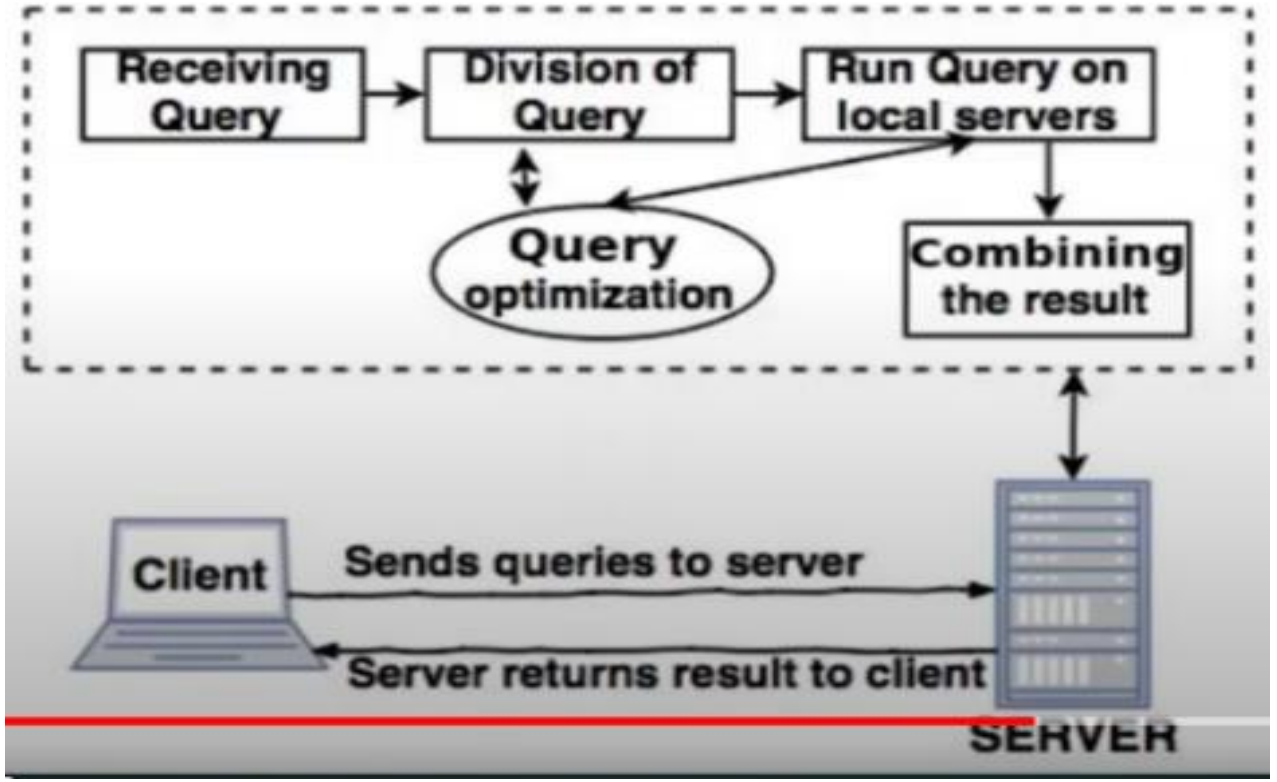
Client-Server Architecture contd..

- The application server approach (indeed, a n-tier distributed approach) can be extended by the introduction of multiple database servers and multiple application servers
- each application server is dedicated to one or a few applications, while database servers operate in the multiple server fashion



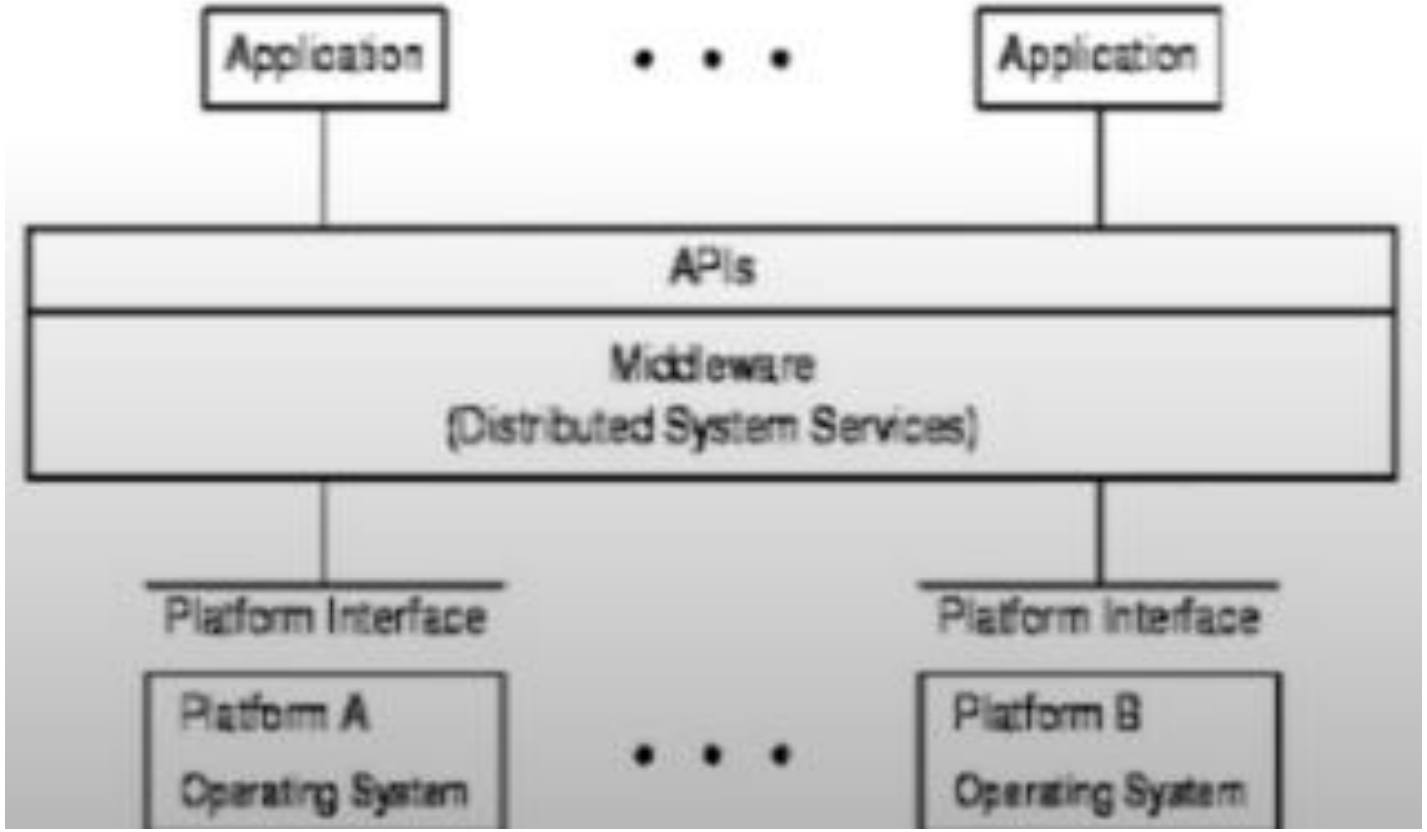
Collaborating Server Architecture

- This architecture is designed to run a single query on multiple servers.
- Specific server breaks the single query into multiple quires and the results are combined and sent back to the client.
- This architecture has a collection of database servers. Each server is capable of executing the current transactions across the database.



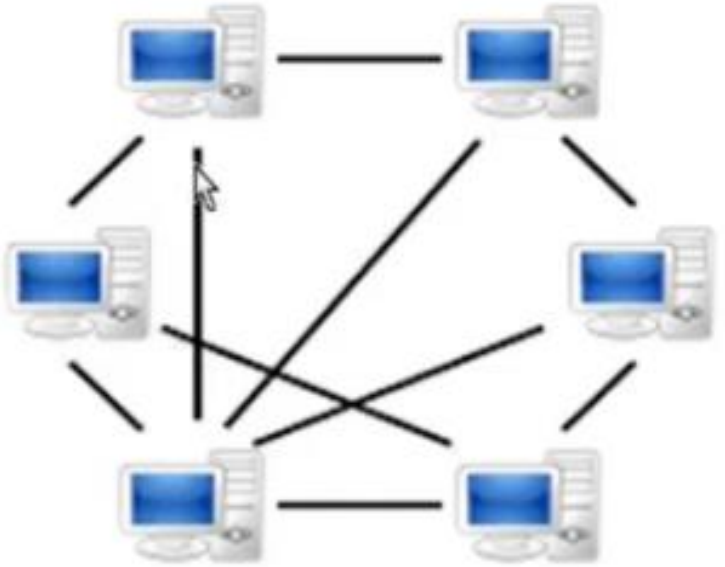
Middleware Architecture

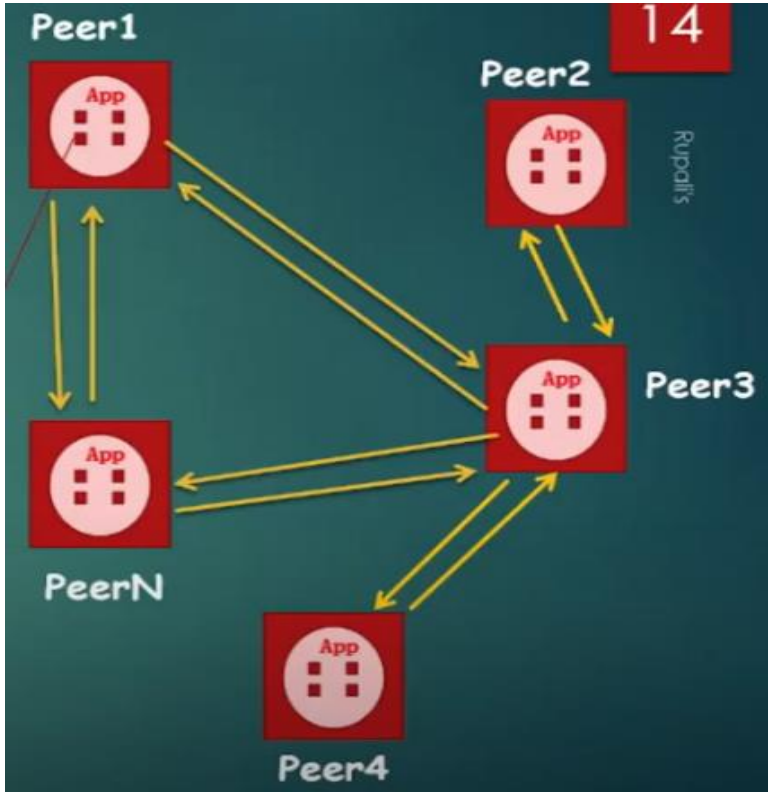
- This is designed in such a way that single query is executed on multiple servers.
- This architecture uses local servers to handle local queries and transactions.
- The middleware software are used for execution of queries & transactions across one or more independent database servers.



Peer-to-Peer Systems

- No specific client or server.
- Each peer acts both as a client and server depending on whether the node is requesting or providing service.
- Each node considered as peer.
- These peers share their resource with other peers and co-ordinate their activities.
- Pattern of communication between the pairs entirely depends on the application.
- Each object is replicated in several systems/sites to distribute the load and manage failure of sites
- Data allocation & retrieval is more complex than client-server systems

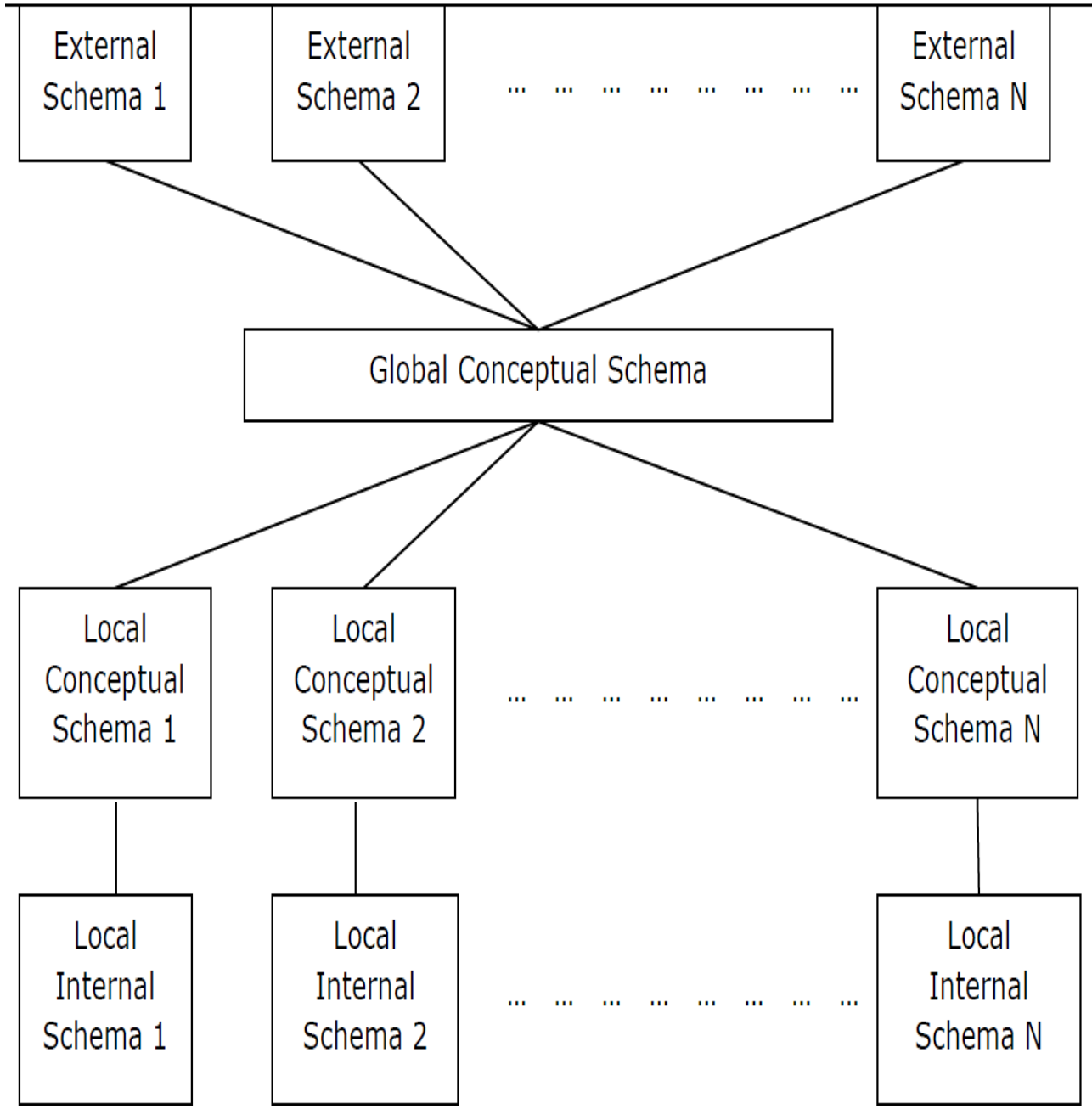




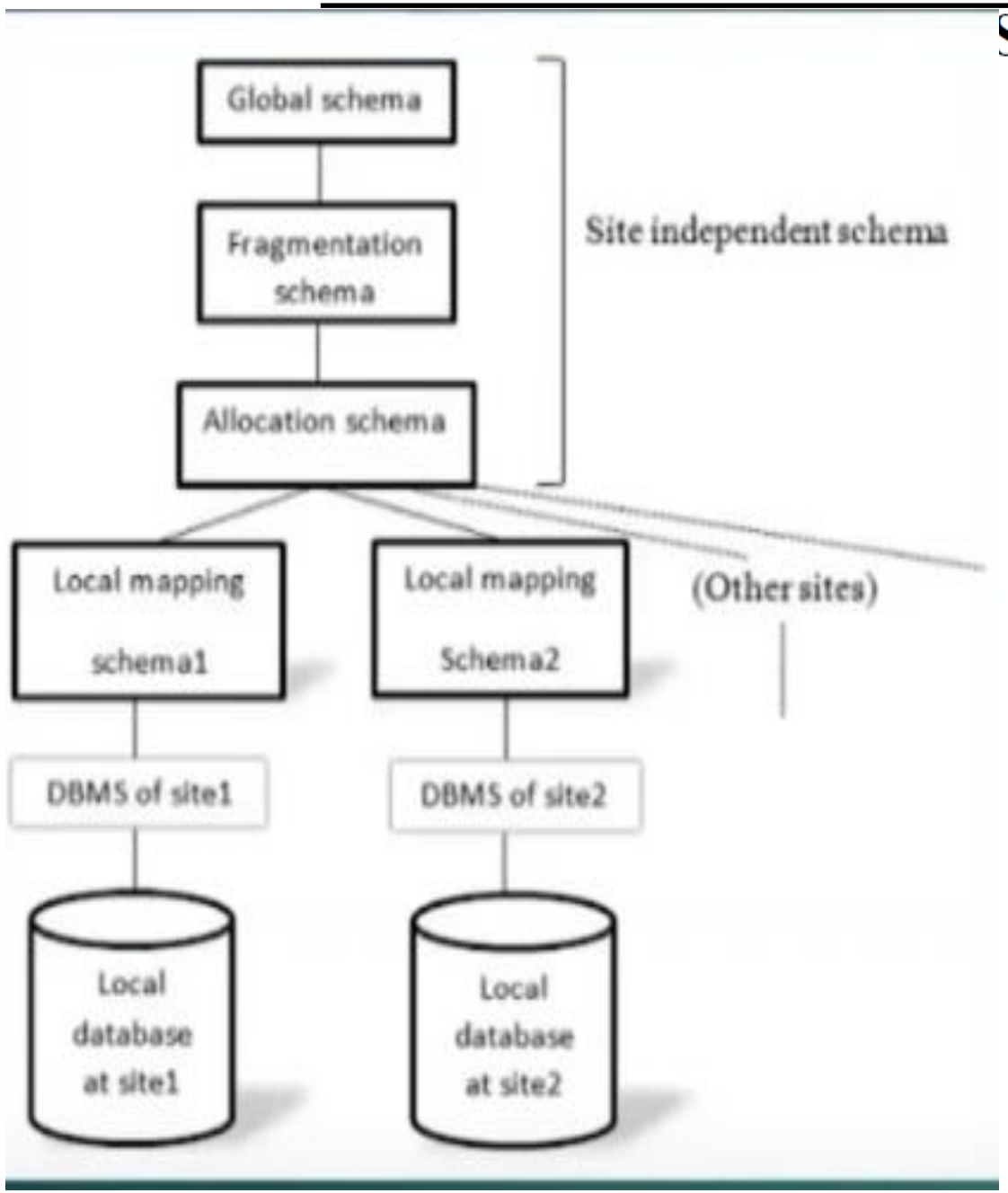
Peer-to-Peer Systems

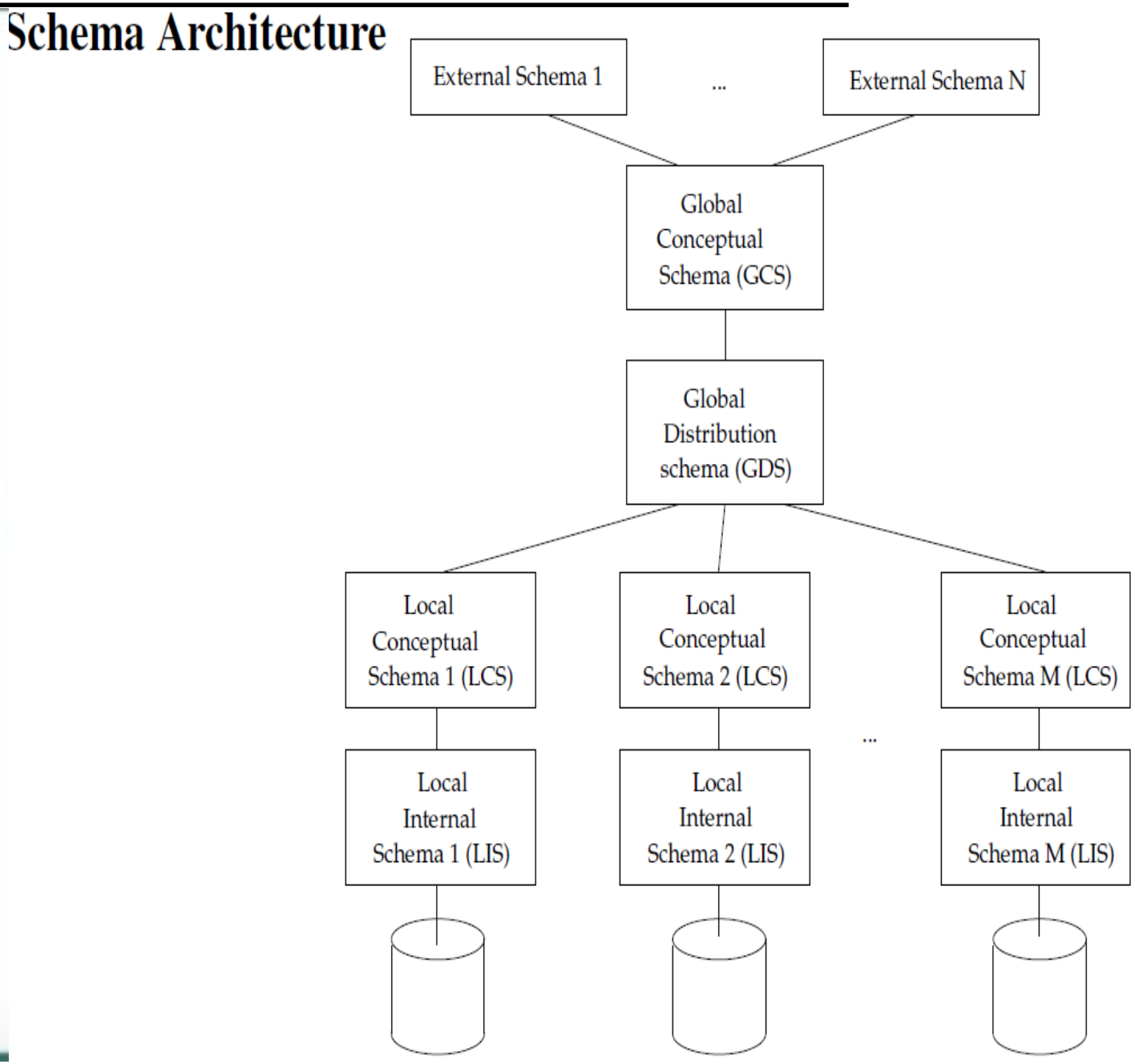
- This architecture generally has four levels of schemas:
- **External Schema:** Depicts user view of data.
- **Global Conceptual Schema:** Depicts the global logical view of data.
- **Local Conceptual Schema:** Depicts logical data organization at each site.
- **Local Internal Schema:** Depicts physical data organization at each site.

contd..



General Reference Architecture of DDBMS

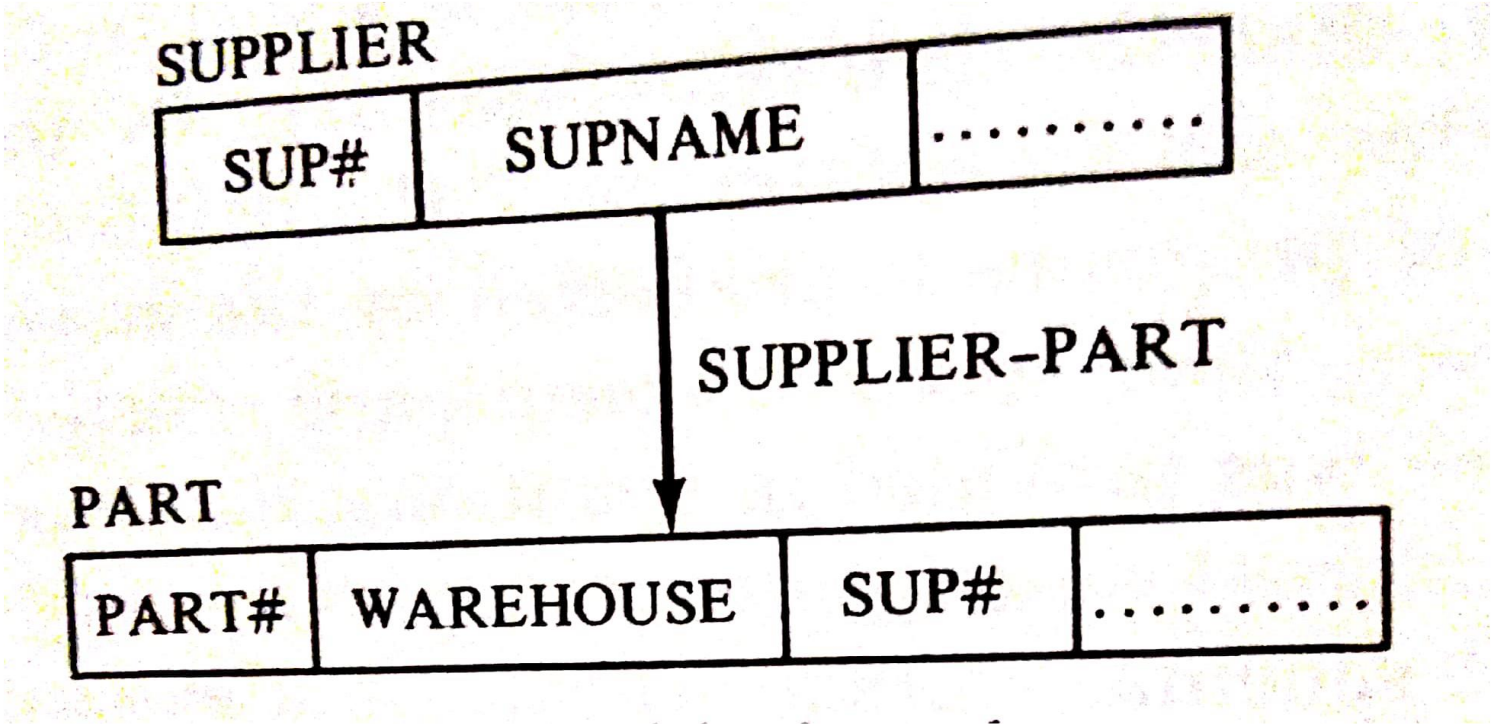




Schema Architecture for peer to peer systems

- **Global schema-** defines data as a whole, not fragmented at all.
- Employee (e_no, e_name, dept_no)
- **Fragmentation Schema-** Specifying the way global relations are fragmented
- Emplyee1 -> dept_no='production' and dept_no=Emplyee2 -> 'sales'
- **Allocation schema** – determining the sites where a particular fragment is allocated, for ex:- Emplyee1 -> at site 1,2 and Emplyee2 -> at site 3,4
- **Local mapping schema –** helps in identifying the global relation schema for any local database relation schema. It is the local mapping schema which facilitates the integration of local database sites into one single global database. It is very much similar to the 3 schema architecture of centralized database.

DISTRIBUTED ACCESS PLAN

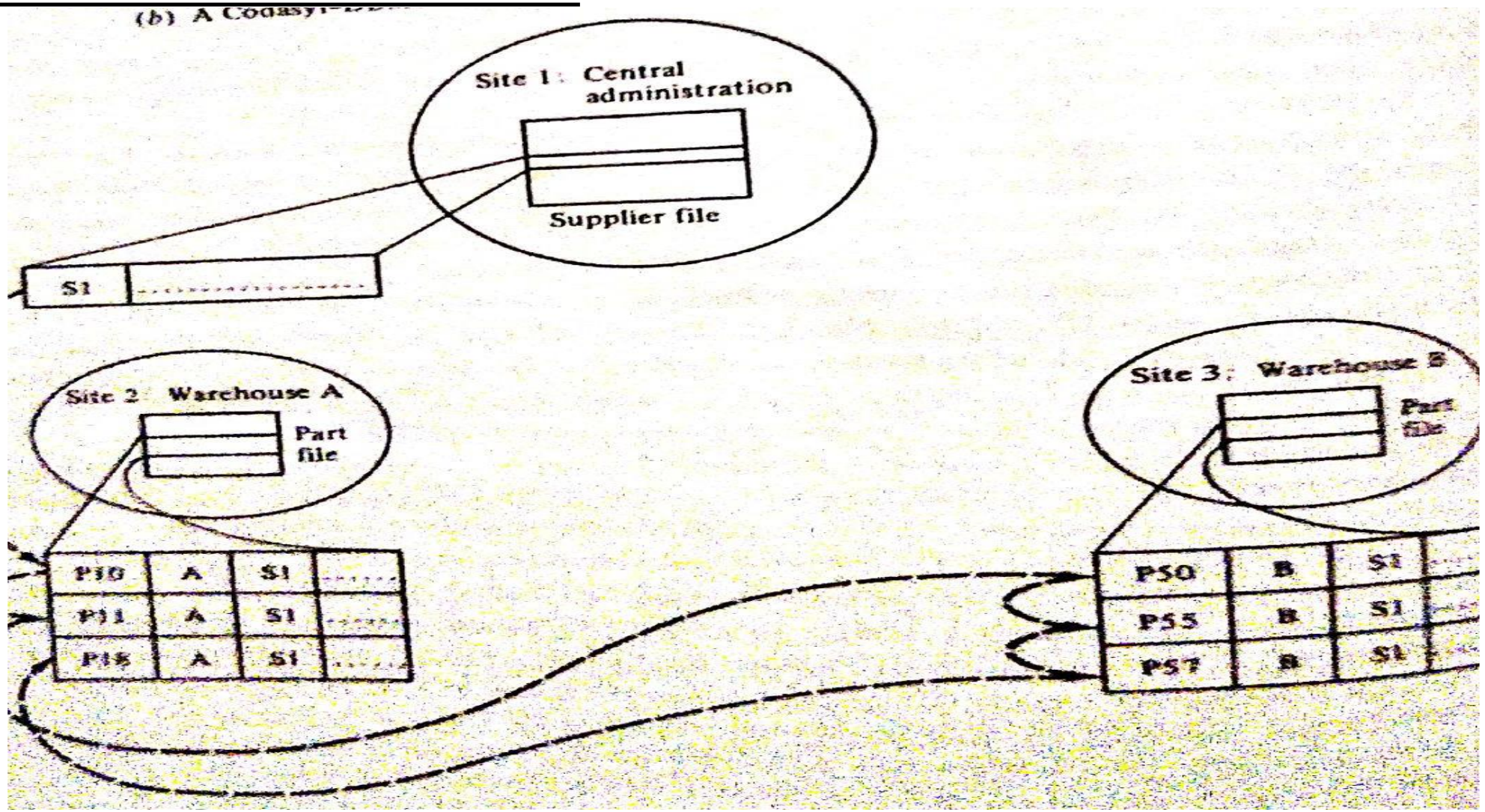


- **Global query:**

SELECT * FROM PART
WHERE SUP# = S1
OR
SELECT * FROM PART, SUPPLIER
WHERE SUP# = S1 AND SUPPLIER.SUP# = PART.SUP#

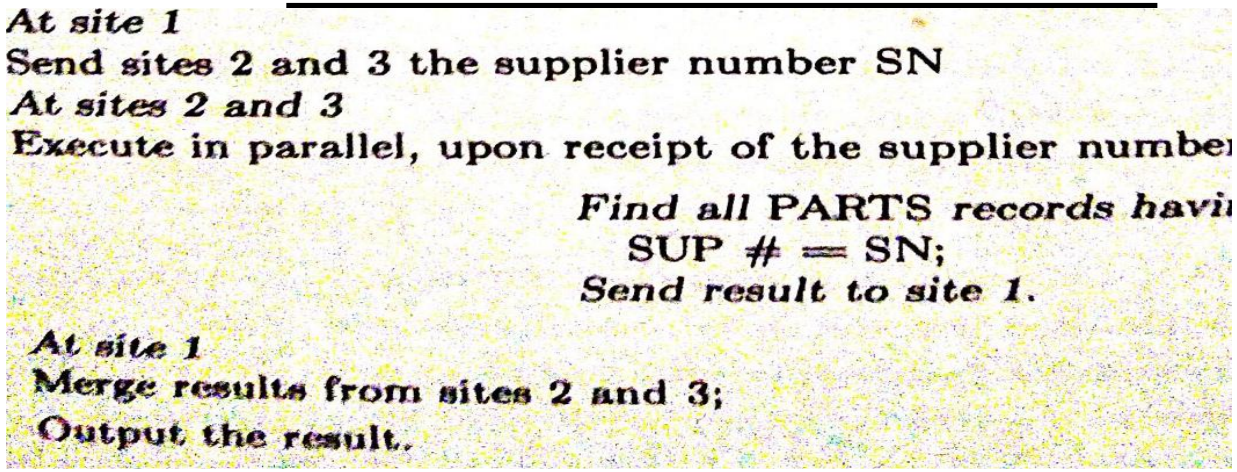
DISTRIBUTED ACCESS PLAN

contd..



- Example:
- Let us assume that, the database is distributed over three sites-
- The supplier file is located at site1(central administration), while the part file is splited into two different subfiles located at site 2 and 3.

DISTRIBUTED ACCESS PLAN



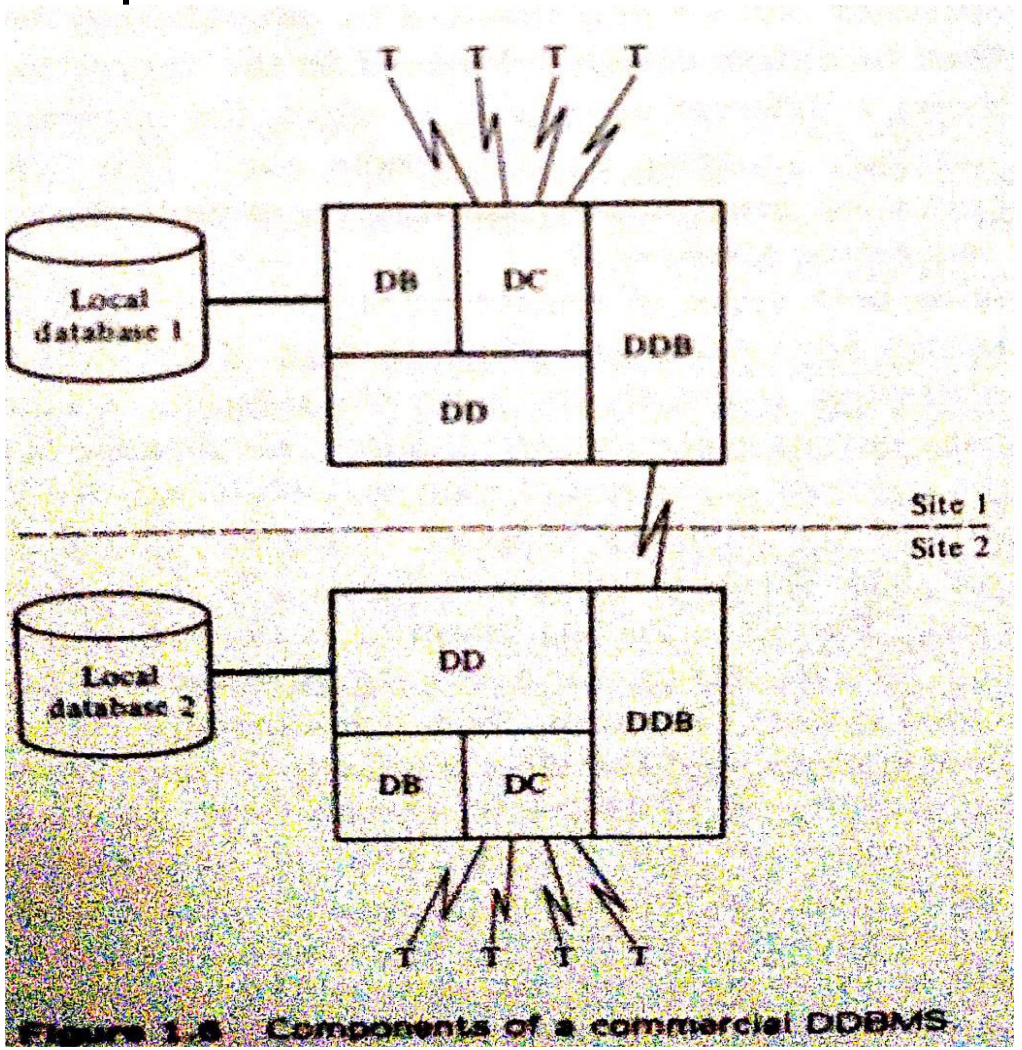
- **Distributed access plan** can be written by the programmer or produced automatically by the optimizer.
- **Global Optimization:-** It determines which data files must be transmitted between sites. The main optimization parameter here is the communication cost, although the cost of accessing the local databases should also be taken into account in some cases. The relative importance of these factors depends on the ratio between communication costs and disk access costs.
- **Local optimization:-** it decides how to perform the local database access at each site; the problems of local optimization implied for traditional non-distributed databases.

Peer-to-Peer Systems contd..

- Physical data organization at each site may be different. Hence each site needs an individual internal schema definition called as local internal schema (LIS).
- The enterprise view of the data is described by global conceptual schema (GCS) and it describes the logical structure of the data at all sites.
- To handle data fragmentation and replication, the logical organization of data at each site needs to be described and therefore, a third layer in the architecture is added known as local conceptual schema (LCS). The global conceptual schema is the union of the local conceptual schemas.
- Finally, user applications and user access to the database is supported by external schemas (ESs), defined as being above the global conceptual schema.
- distributed DBMS translates global queries into a group of local queries, which are executed by distributed DBMS components at different sites that communicate with one another

SOFTWARE COMPONENT OF DDBs

1. The database management component (DB)
2. The data communication component (DC)
3. The data dictionary (DD), which is extended to represent information about the distribution of data in the network
4. The distributed database component (DDB)



Components of a commercial DDBMS.

Components of a Distributed DBMS

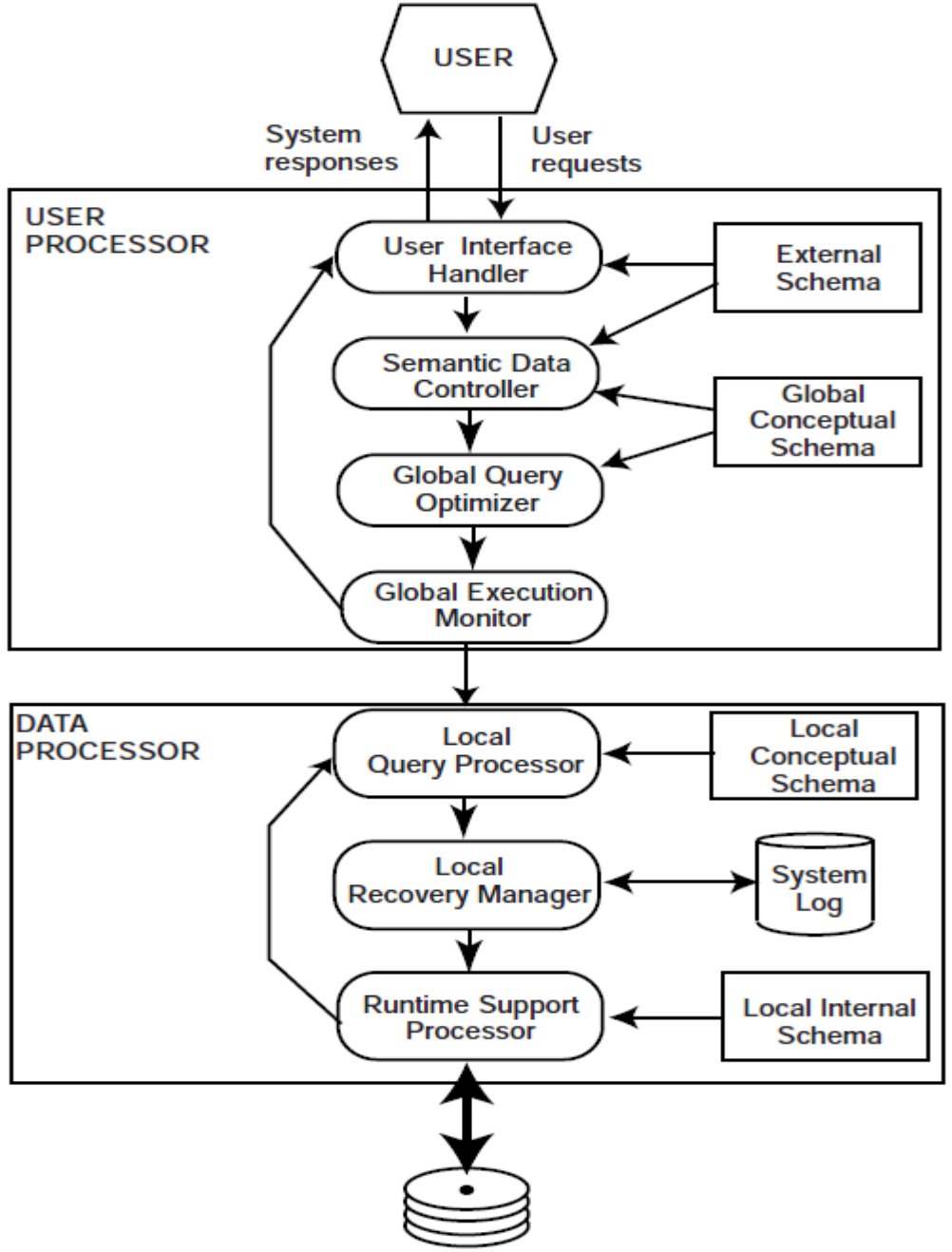
User Processor- handles interactions with user.

1. user interface handler- interprets the user commands and also formats the result data sent back to the user.

2. semantic data controller- checks the integrity constraints and authorizations that are defined in global conceptual schema.

3. global query optimizer and decomposer- determines an execution strategy to minimize the cost and translates the global queries into local ones.

4. Global execution monitor- coordinates distributed execution of user request. This part is also called as distributed transaction manager. Execution monitors at various sites usually communicate with each other while executing queries.



Components of a Distributed DBMS contd..

- **Data processor –** deals with data storage and having the following parts-

1. local query processor- responsible for choosing the best access path to access any data item. The term access path refers to the data structures and algorithms used to access the data. A typical access path, for example, is an index on one or more attributes of a relation

2. Local recovery manager- responsible for making sure that the local database remains consistent even when failures occurs

3. run-time support processor- physically accesses the database according to the physical commands generated by the query optimizer.

- The run-time support processor is the interface to the operating system
- It contains the database buffer (or cache) manager, which is responsible for maintaining the main memory buffers and managing the data accesses.

N.B- In peer-to-peer systems, both the user processor modules and the data processor modules should be present in each machine

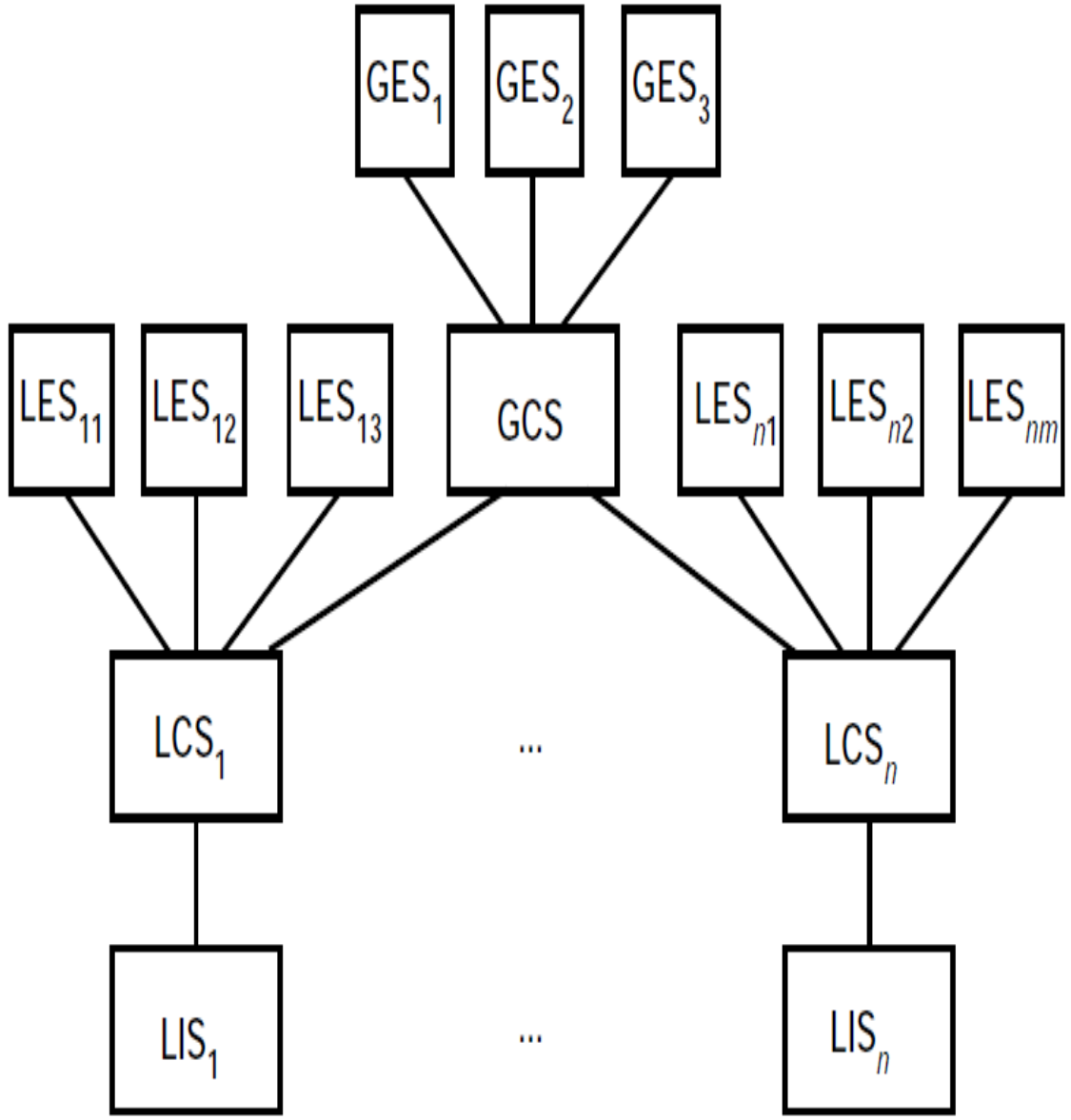
Multi - DBMS Architectures

- **Multidata base systems (MDBS)-** individual DBMSs (whether distributed or not) are fully autonomous and having no cooperation with each other. They may not even "know" of each other's existence or how to talk to each other.

➢In case of logically integrated distributed DBMS, GCS is the conceptual view of entire database, while in distributed MDBMS, it is the collection of some of the local DBMSs that are sharable.

➢So, global database in logically integrated DDBMS is actually equal to the union of local databases, whereas in multi-DDBMS it is only a subset of the same union.

➢Hence GCS in multi-DDBMSs is the mapping from local conceptual schemas to a global schema, however for logically integrated DDBMS the mapping is in the reverse direction



16 MDBS Architecture with a GCS

Multi - DBMS Architectures contd..

This is an integrated database system formed by a collection of two or more autonomous database systems and are expressed through six levels of schemas:

1. **Multi-database View Level:** Depicts multiple user views comprising of subsets of the integrated distributed database.

2. Multi-database Conceptual Level: Depicts integrated multi-database that comprises of global logical multi-database structure definitions.

3. Multi-database Internal Level: Depicts the data distribution across different sites and multi-database to local data mapping.

4. Local database View Level: Depicts public view of local data.

5. Local database Conceptual Level: Depicts local data organization at each site.

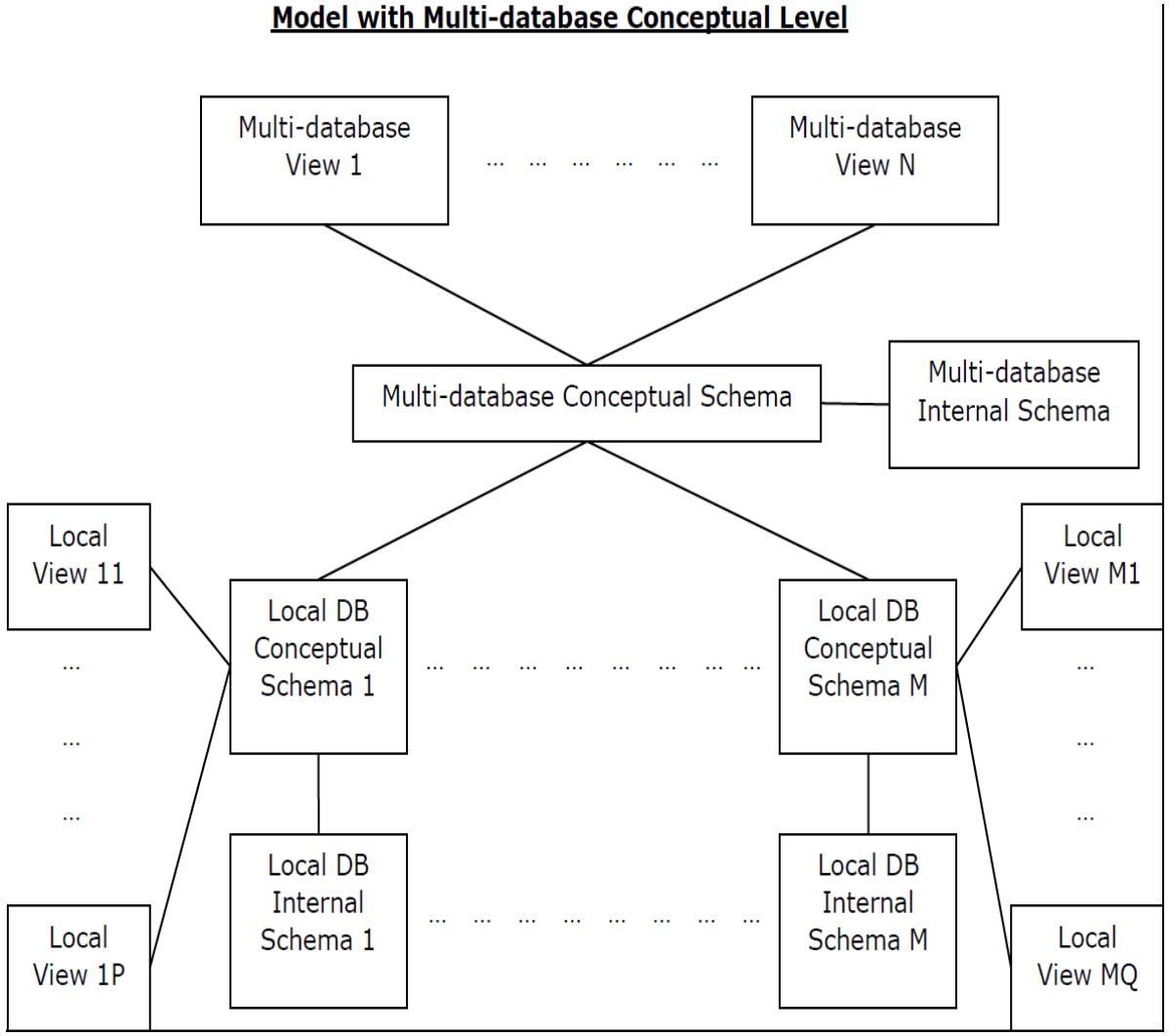
6. Local database Internal Level: Depicts physical data organization at each site.

Multi - DBMS Architectures contd..

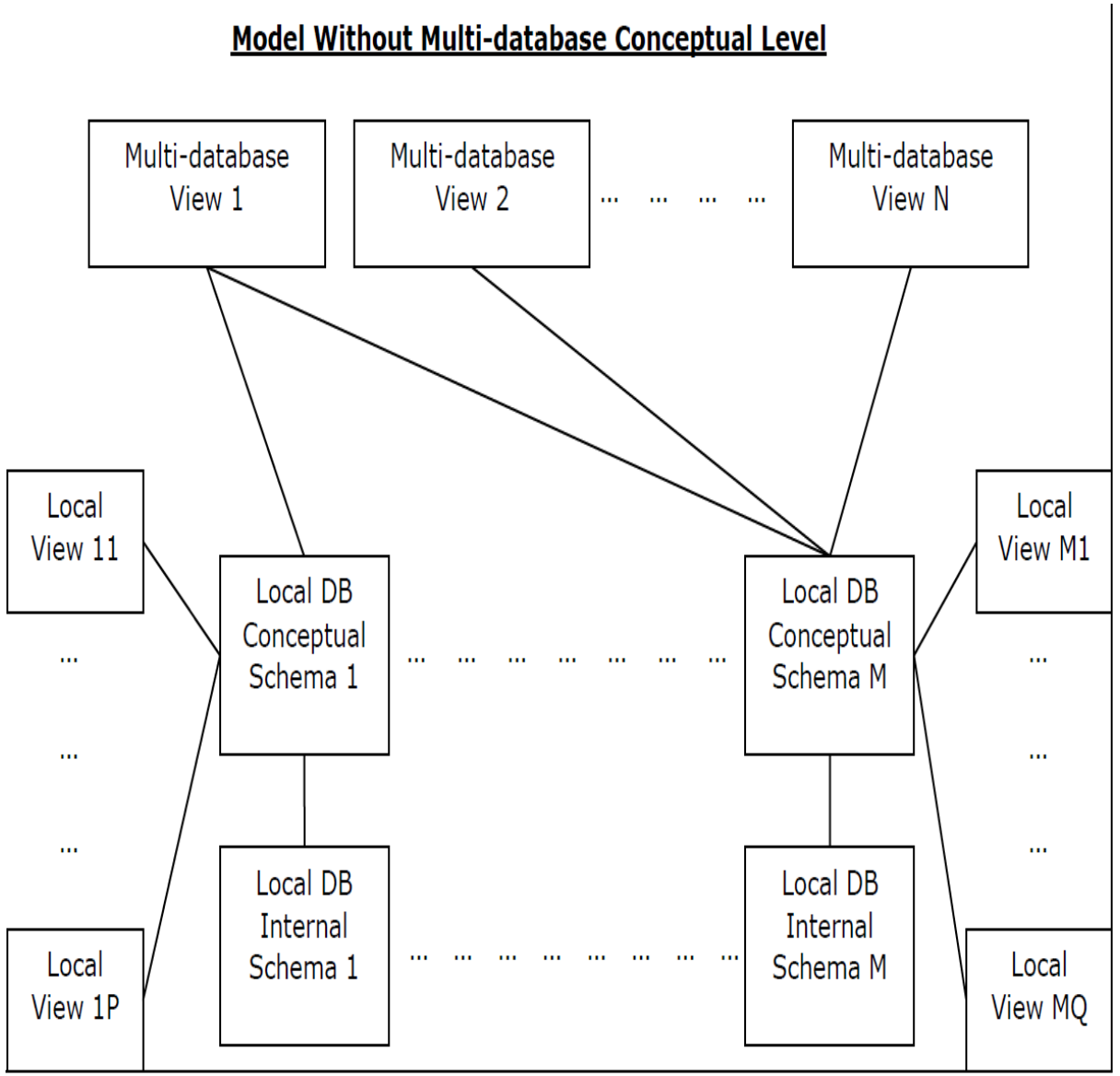
There are two design alternatives for multi-DBMS:

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

Model with Multi-database Conceptual Level



Model Without Multi-database Conceptual Level



Conceptual Schema Definition

RELATION EMP [
    KEY = {ENO}
    ATTRIBUTES = {
        ENO : **CHARACTER**(9)
        ENAME : **CHARACTER**(15)
        TITLE : **CHARACTER**(10)
    }
]
RELATION PAY [
    KEY = {TITLE}
    ATTRIBUTES = {
        TITLE : **CHARACTER**(10)
        SAL : **NUMERIC**(6)
    }
]

Internal Schema Definition

```
RELATION EMP [
        KEY = {ENO}
        ATTRIBUTES = {
                ENO         :  CHARACTER(9)
                ENAME       :  CHARACTER(15)
                TITLE       :  CHARACTER(10)
        }
]
```

```
INTERNAL_REL EMPL [
        INDEX ON E# CALL EMINX
        FIELD = {
                HEADER      : BYTE(1)
                E#  : BYTE(9)
                ENAME       : BYTE(15)
                TIT : BYTE(10)
        }
]
```

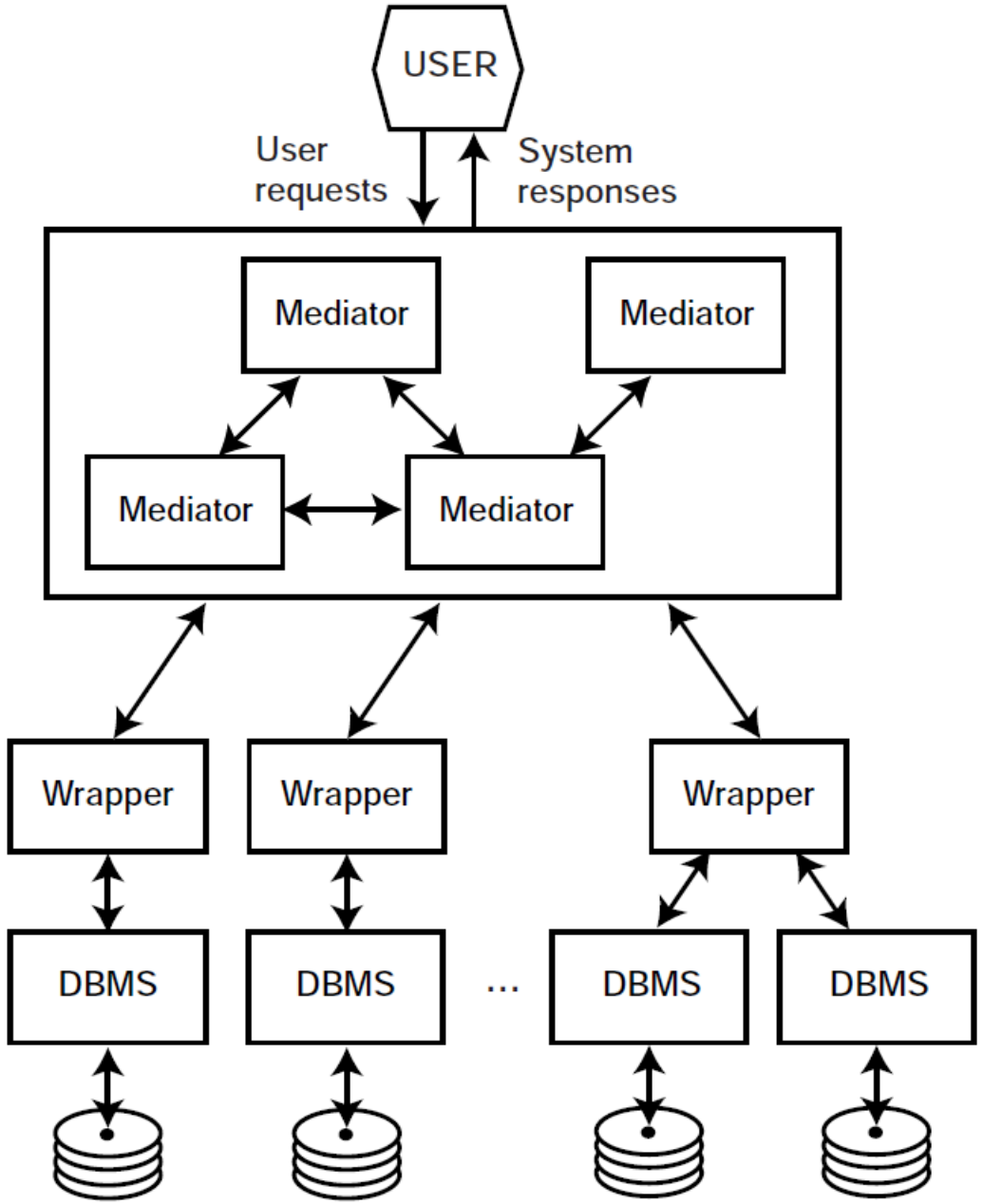
External View Definition – Example 1

Create a BUDGET view from the PROJ relation

```
CREATE    VIEW      BUDGET(PNAME, BUD)
 AS       SELECT    PNAME, BUDGET
           FROM     PROJ
```

Mediator Architecture for MDBMS

- MDBS provides a layer of software that runs on top of these individual DBMSs (mediator) and provides users the facilities of accessing various databases.
- The multi-DBMS layer may run on multiple sites or there may be central site where those services are offered.
- Mediator level implements the GCS for handling user queries. Mediators typically operate using a common data model and interface language.
- To deal with potential heterogeneities, wrappers are implemented whose task is to provide a mapping between a source DBMSs view and the mediators' view. For example, if source DBMS is relational, but the mediator implementations are object-oriented, the required mappings are established by the wrappers.



Data Fragmentation, Replication & Distribution Issues

- Two basic alternatives for placing the data: partitioned (or non-replicated) and replicated.
- In partitioned scheme the database is divided into a number of disjoint partitions each of which is placed at a different site.
- Replicated designs can be either fully replicated (also called fully duplicated) where the entire database is stored at each site, or partially replicated (or partially duplicated) where each partition of the database is stored at more than one site, but not at all the sites.

Distributed Directory Management

A directory contains information (such as descriptions and locations) about data items in the database. A directory may be global to the entire DDBMS or local to each site; it can be centralized at one site or distributed over several sites; i.e there can be a single copy or multiple copies.

Directory Concept in DDBMS

The distributed database schema information is needed during distributed query optimization. The schema information is stored as meta data in a data dictionary/directory, In the case of DDBMS, the schema definition is done both at the global level (i.e. GCS) and at the local sites (i.e. LCS). Hence, there are two types of directories:

(i) a global directory/dictionary (GD) that describes the database schema for the end users and it permits mapping between external schemas and GCS.

(ii) the local directory/dictionary (LD), that describes the local mappings from GCS and also the local schema at each site.

As the directory is itself a database containing metadata about the actual data, so DDB design concept also applies to the directory management.

- Hence, a directory may be either global or local to each site. In other words, there might be a single directory containing information about all the database, or a number of directories, each containing information stored at a particular site.

Directory Concept in DDBMS

- In case of global directory, it may be maintained centrally at one site, or by distributing it over a number of sites.
- Keeping the directory at one site might increase the load at that site i.e bottleneck.
- Distributing it over a number of sites, on the other hand, increases the complexity of managing such directories.
- Also, there may be a single copy of the directory or multiple copies in different sites.
- Multiple copies would provide more reliability, in accessing the directory.
- On the other hand, keeping the directory up to date would be considerably more difficult, since multiple copies would need to be updated.
- Therefore, the choice should depend on the environment in which the system operates and should be made by balancing all these factors.