BELGIUM CAMPUS iTversity

It's the way we're wired

DATABASE DEVELOPMENT

ARE YOU READY?

DBD371/381

DISTRIBUTED DATABASES

Distributed Database Architectures

LEARNING OBJECTIVES

BELGIUM CAMPUS iTversity
It's the way we're wired

- ANSI/SPARC Architecture
- Generic Centralized DBMS Architecture
- Architectural Models for Distributed DBMSs
- Architectural Alternatives

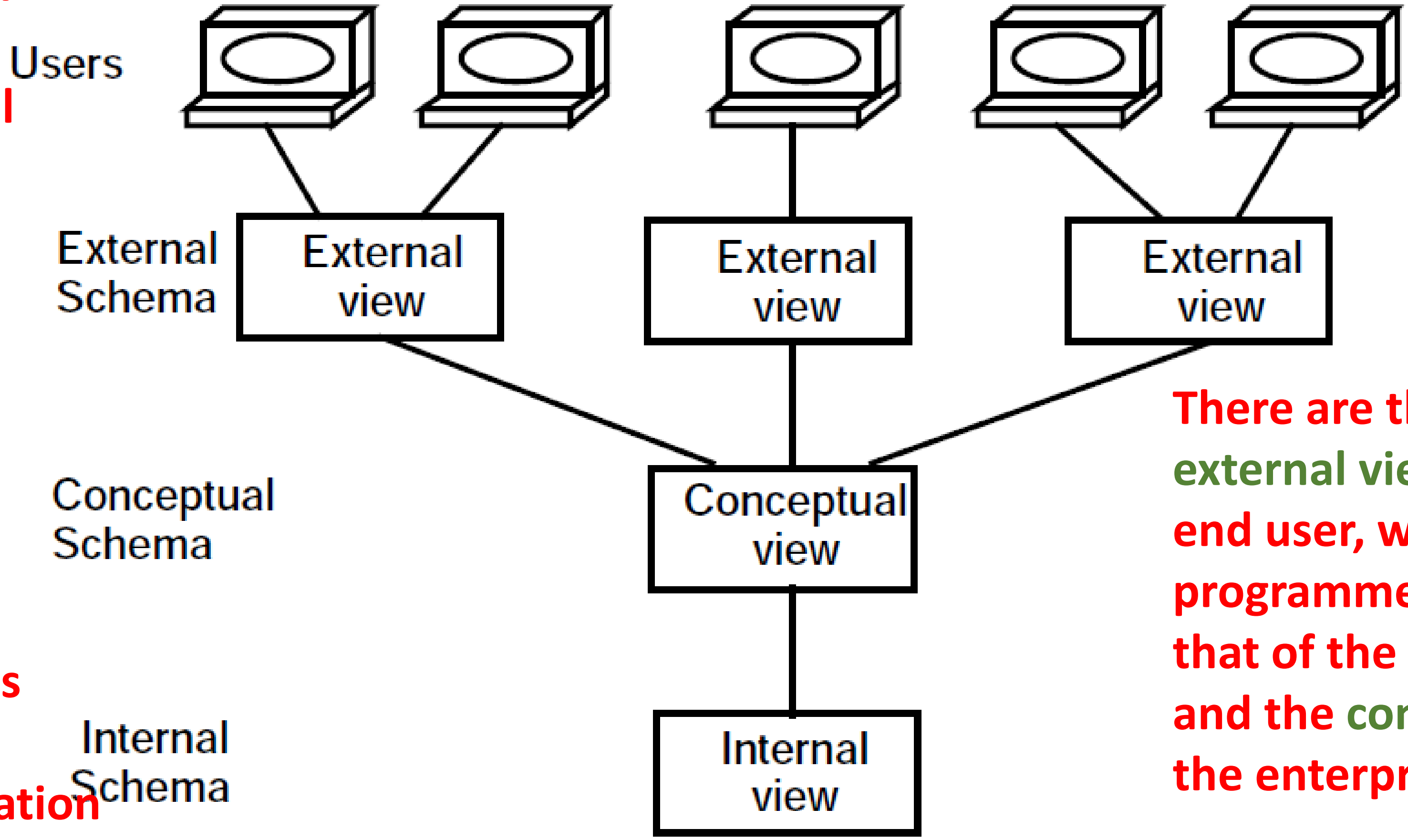
Distributed DBMS Architecture

- The architecture of a system defines its structure.
- Generic architecture of a centralized DBMSs
    - ❑Three “reference” architectures for a distributed DBMS exist:
        1. client/server systems,
        2. peer-to-peer distributed DBMS
        3. multidatabase systems
- Datalogical DBMS architecture→ Focuses on the different user classes and roles and their varying views on data

BELGIUM CAMPUS iTversity
It's the way we're wired

ANSI/SPARC Architecture

The separation of the external schemas from the conceptual schema enables logical data independence,



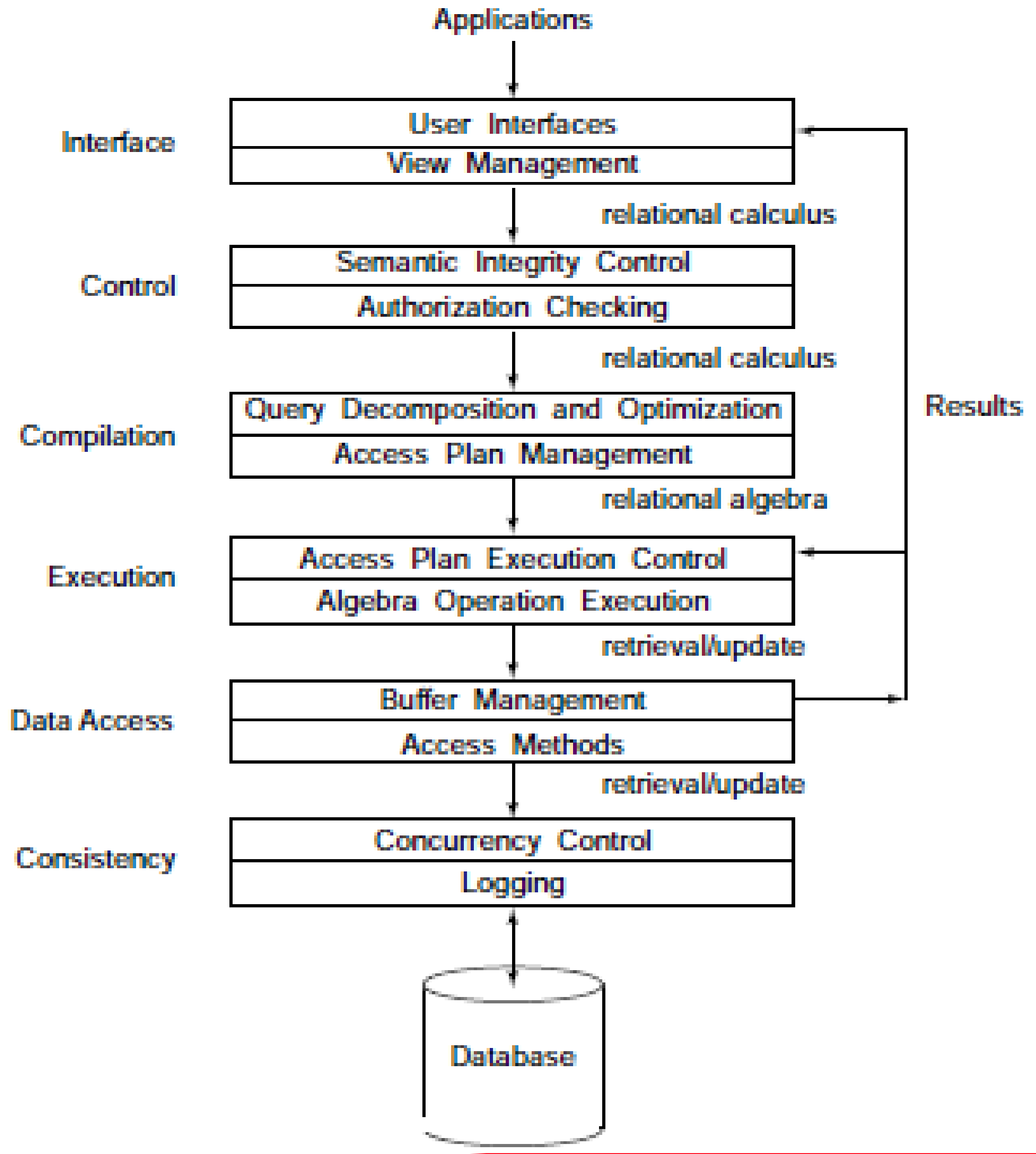
There are three views of data: the external view, which is that of the end user, who might be a programmer; the internal view, that of the system or machine; and the conceptual view, that of the enterprise

the internal view, deals with the physical definition and organization of data

Generic Centralized DBMS Architecture

- The functions performed by a DBMS can be layered
  1. **interface layer** manages the interface to the applications.
  2. **control layer** controls the query by adding semantic integrity predicates and authorization predicates.
  3. **query processing (or compilation) layer** maps the query into an optimized sequence of lower-level operations. This layer is concerned with performance
  4. **execution layer** directs the execution of the access plans, including transaction management (commit, restart) and synchronization of algebra operations
  5. **data access layer** manages the data structures that implement the files, indices,etc.
  6. **consistency layer** manages concurrency control and logging for update requests.

A Generic Centralized DBMS Architecture



ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- The DDBMS architecture can be discussed from three different dimensions.
  1. **Autonomy:** Degree to which member databases can operate independently
  2. **Distribution:** Data is distributed physically on multiple sites/machines.
  3. **Heterogeneity:** Simply means variation / difference.

Autonomy

- Autonomy, refers to the distribution of control, not of data.
- Autonomy is the Degree to which member databases can operate independently Member databases are those database that constitute the distributed database system.
- Autonomy is how much independently the member databases perform their functionality or make their decision.
- **Requirements of an autonomous system:**
  - ➢The local operations of the individual DBMSs are not affected by their participation in the distributed system.
  - ➢The manner in which the individual DBMSs process queries and optimize them should not be affected by the execution of global queries that access multiple databases.
  - ➢System consistency or operation should not be compromised when individual DBMSs join or leave the distributed system.

AUTONOMY

- Three autonomies:
  1. **Communication Autonomy:**
  2. **Design autonomy**
  3. **Execution Autonomy :**

Autonomy

- **Dimensions of autonomy:**
- 1. **Design autonomy:**
  - ➢ Individual DBMSs are free to use the data models and transaction management techniques that they prefer.
  - ➢ Freedom for designer to make decision related to design (data model. DBMS, machine to buy, operating system)
- 2. **Communication autonomy:**
  - ➢ Each of the individual DBMSs is free to make its own decision as to what type of information it wants to provide to the other DBMSs or to the software that controls their global execution.
  - ➢ What part of data we want to share, depends on the nature of bond/coupling between member databases
- 3. **Execution autonomy:**
  - ➢ Each DBMS can execute the transactions that are submitted to it in any way that it wants to.
  - ➢ Decisions on and the authority for when to share the data when to shut down etc. in case of same organization, the rules are formulated at the head office level and all member databases have to follow that, otherwise they are autonomous

BELGIUM CAMPUS iTversity
It's the way we're wired

Autonomy Classifications

- **Alternatives:**

1. **Tight integration** where a single-image of the entire database is available to any user who wants to share the information, which may reside in multiple databases. All users get a single common view with a feeling that there are no component databases.
2. **semiautonomous** systems that consist of DBMSs that can (and usually do) operate independently, but have decided to participate in a federation to make their local data sharable. Component database are present and they relate their data and at certain time they share their data to form a common view.
3. T**otal isolation,** where the individual systems are stand-alone DBMSs that know neither of the existence of other DBMSs nor how to communicate with them.

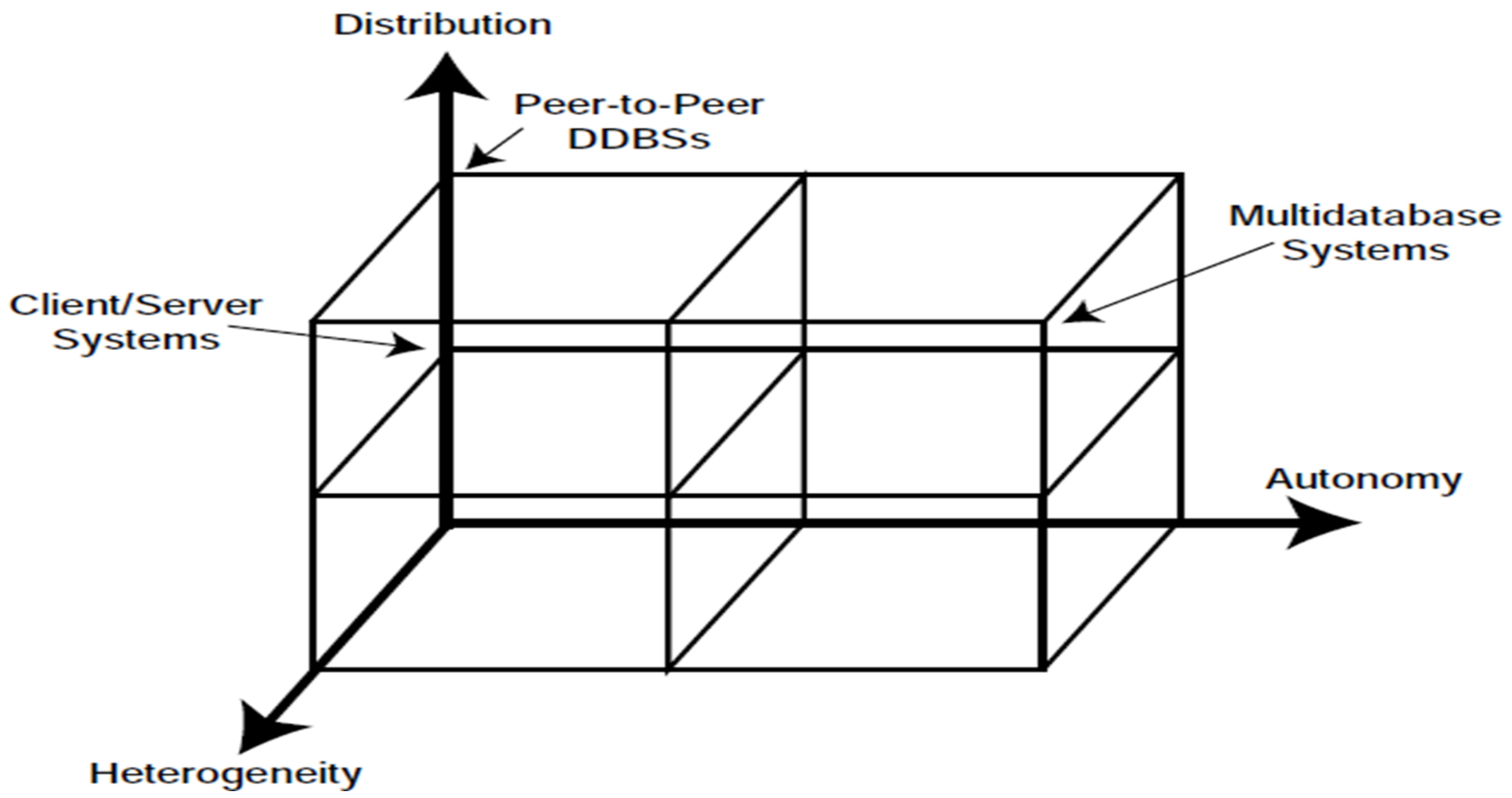
Distribution

- A system cannot be called truly a DDBMS if the data is not distributed physically.
- The distribution could be of data or of control;
- The distribution dimension of the taxonomy deals with data.
- **Two major architectures are proposed**
  1. **client/server distribution:** concentrates data management duties at servers while the clients focus on providing the application environment including the user interface (client and server, both are machines not processes)
  2. **peer-to-peer distribution (or full distribution).:**Each machine has full DBMS functionality and can communicate with other machines to execute queries and transactions. (servers performing independently and cooperating with each other.)

Heterogeneity

- Heterogeneity may occur in various forms in distributed systems, ranging from hardware heterogeneity and differences in networking protocols to variations in data managers.
- Heterogeneity Simply means variation / difference.
- May be in different form
- Different component method must be compatible.
- **Semantic Heterogeneity:** Different component databases representing same real-world concept differently. It is most difficult to handle.
- **Heterogeneity relate to**
    - ➢data models,
    - ➢query languages,
    - ➢transaction management protocols.

DIMENSIONS FOR ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS



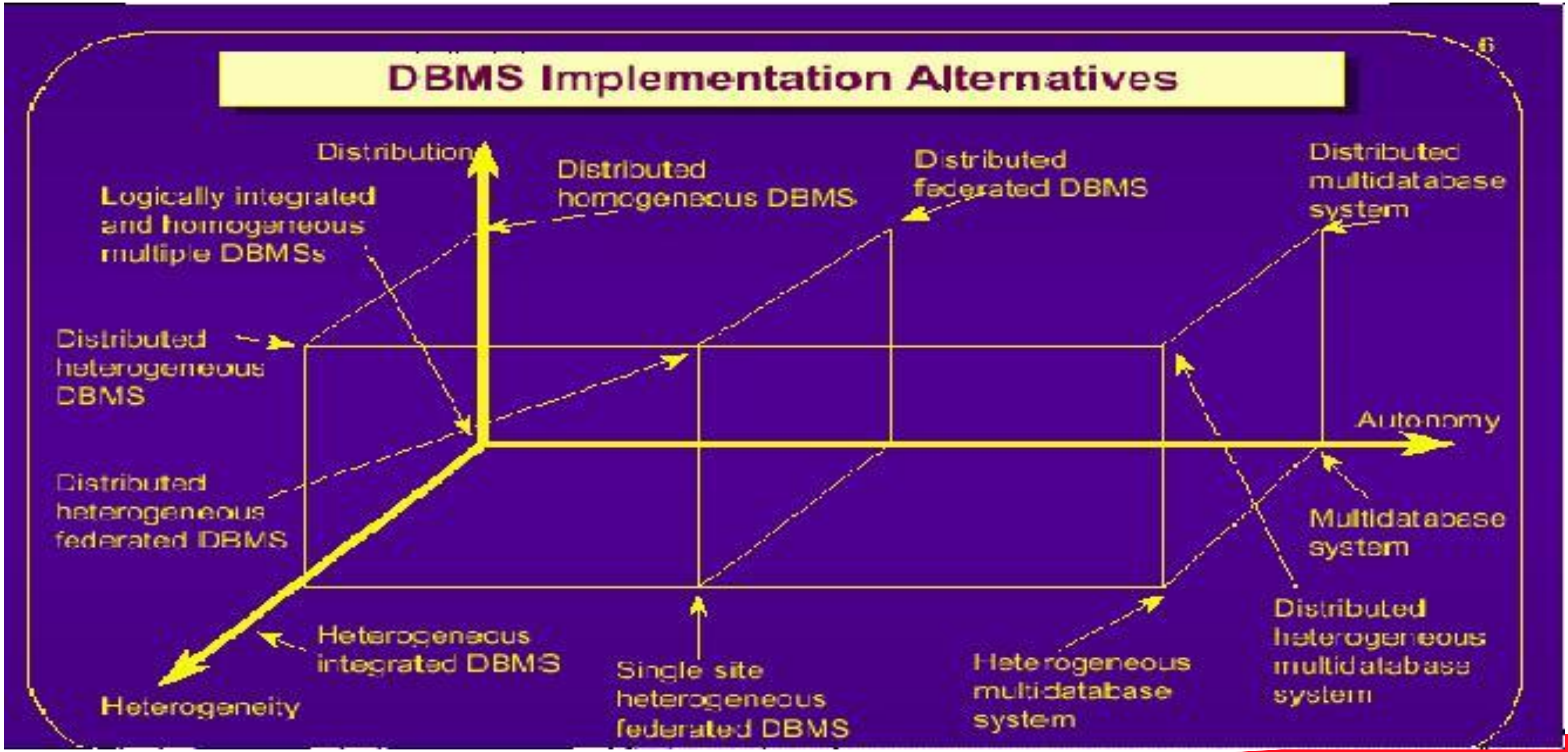
ARCHITECTURAL ALTERNATIVE

- Dimensions may exist at different levels in different architectures as given below:
- **Autonomy (0,1,2)**
  - 0-Tight hydrogenation
  - 1-semi autonomous
  - 2-total isolation
- **Distribution (0,1,2)**
  - 0→No distribution
  - 1 →client/server
  - 2 →Peer to Peer
- **Heterogeneity (0,1)**
  - 0→homogeneous
  - 1-->Heterogeneous

ARCHITECTURAL ALTERNATIVE

- Different combinations of the dimensions give rise to different architectures of DBMS:
- Example
- (A1,D2,H1)
- A1(semi autonomous) D2 (Peer to Peer) H1 (Heterogeneity)
- All possible combinations are shown in the diagram below:

ARCHITECTURAL ALTERNATIVE



6
DBMS Implementation Alternatives
Distribution
Logically integrated and homogeneous multiple DBMSs
Distributed homogeneous DBMS
Distributed federated DBMS
Distributed multidatabase system
Distributed heterogeneous DBMS
Autonomy
Distributed heterogeneous federated DBMS
Multidatabase system
Distributed heterogeneous multidatabase system
Heterogeneous integrated DBMS
Single site heterogeneous federated DBMS
Heterogeneous multidatabase system
Heterogeneity

The Architectural Alternatives

1. Client/Server Systems
2. Peer-to-Peer Systems
3. Multidatabase System Architecture

Client/Server Systems

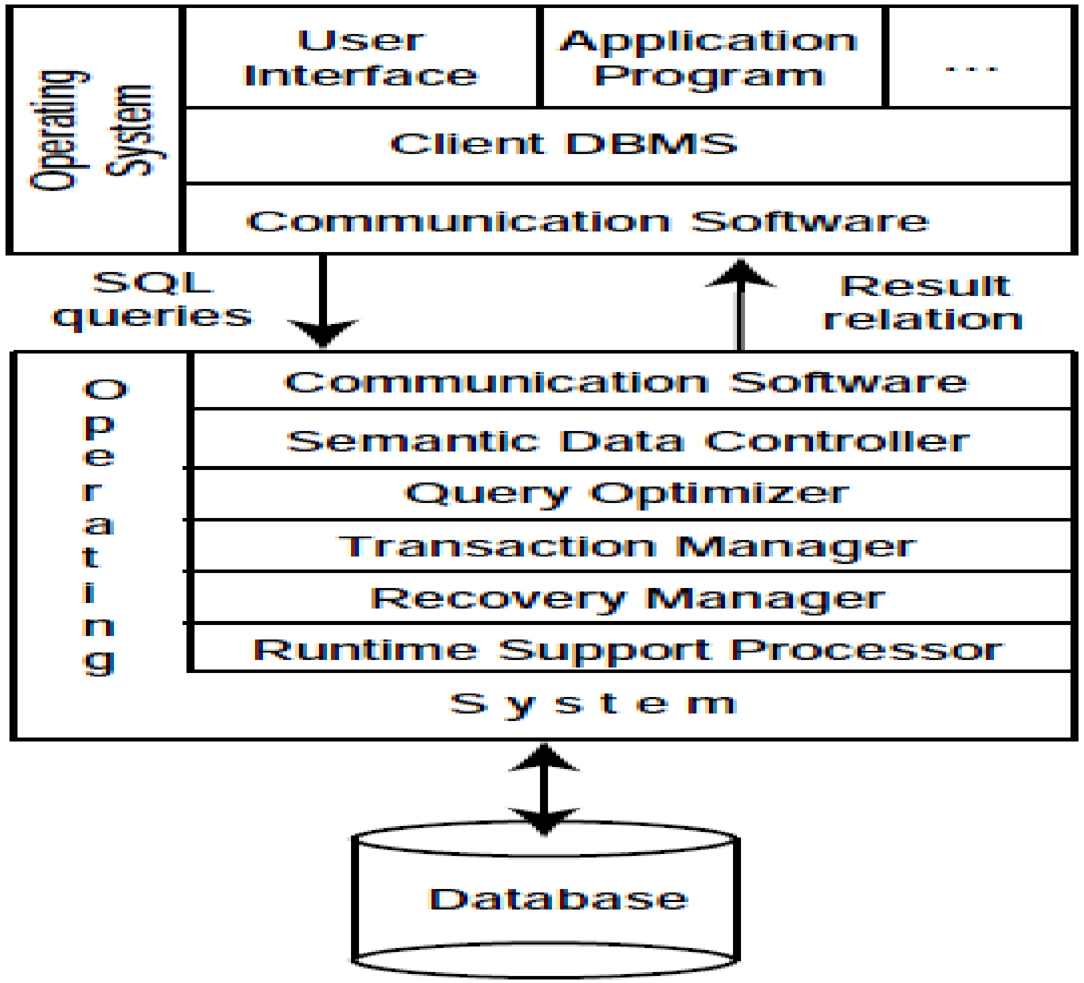
- The general idea distinguishes the functionality that needs to be provided and divide these functions into two classes: server functions and client functions.
- Provides a two-level architecture which makes it easier to manage the complexity of modern dbmss and the complexity of distribution.
- The functionality allocation between clients and serves differ in different types of distributed DBMSs
  - ➢ In relational systems, the server does most of the data management work where query processing and optimization, transaction management and storage management is done at the server.
- The client, provides the application and user interface, and has a DBMS client module that is responsible for managing the data that is cached to the client and (sometimes) managing the transaction locks that may have been cached.
- consistency checking of user queries may also be done at the client side, though not common since it requires the replication of the system catalog at the client machines
- operating system and communication software runs on both the client and the server

Types Of Client/Server Architecture

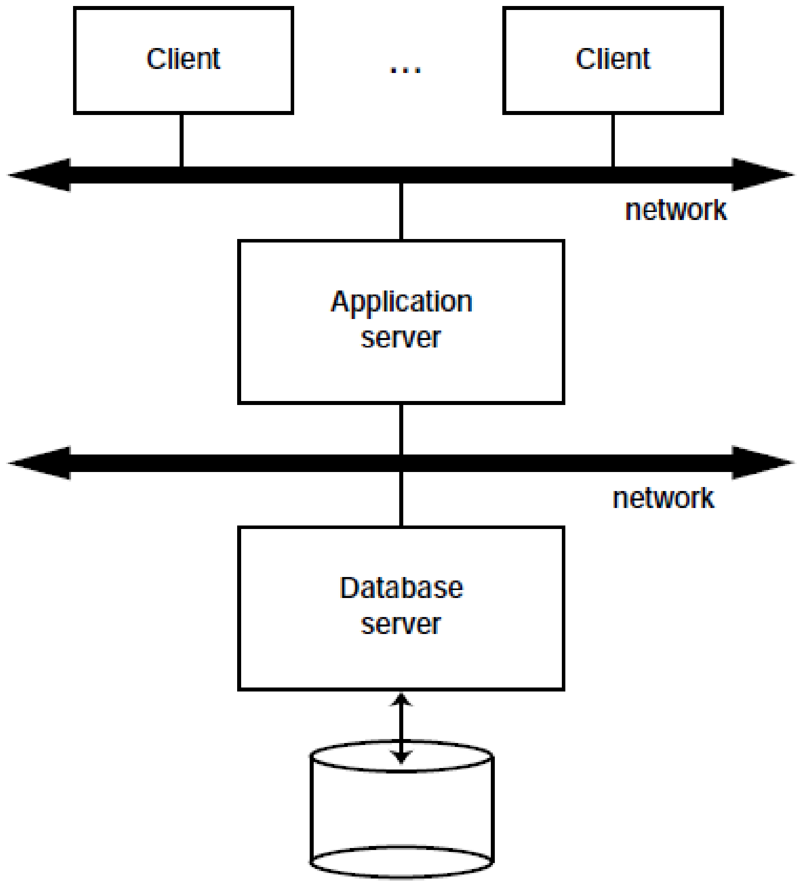
- **Multiple client/single server:** one server accessed by multiple clients
- **Multiple client/multiple server:** Two alternative management strategies are possible
  1. Each client manages its own connection to the appropriate server
     --system loads the client machines with additional responsibilities. ("heavy client")
  2. Each client knows of only its "home server" which then communicates with other servers as required. --concentrates the data management functionality at the servers("light clients.")

Architectural Alternatives

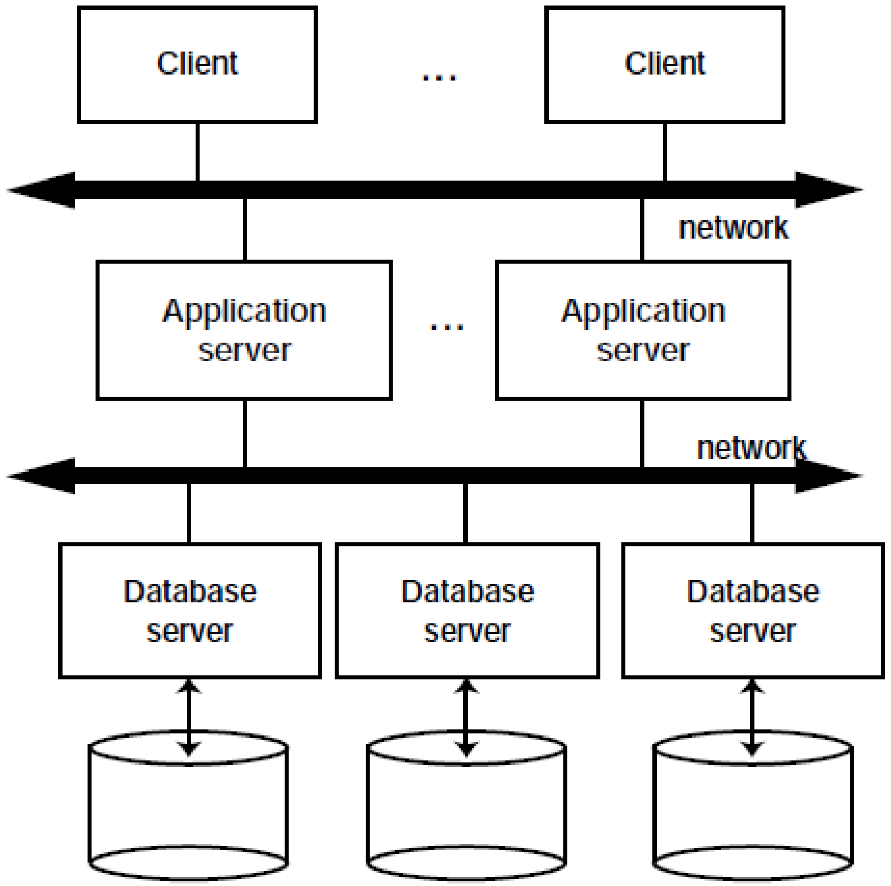
- Client/Server Systems



Database Server Approach



Distributed Database Servers

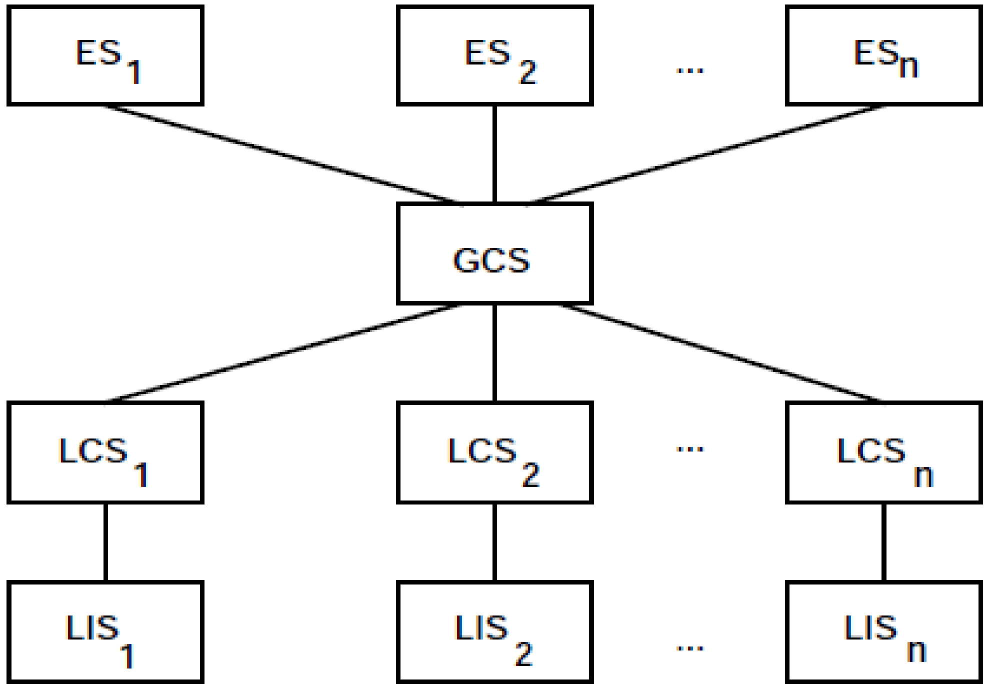


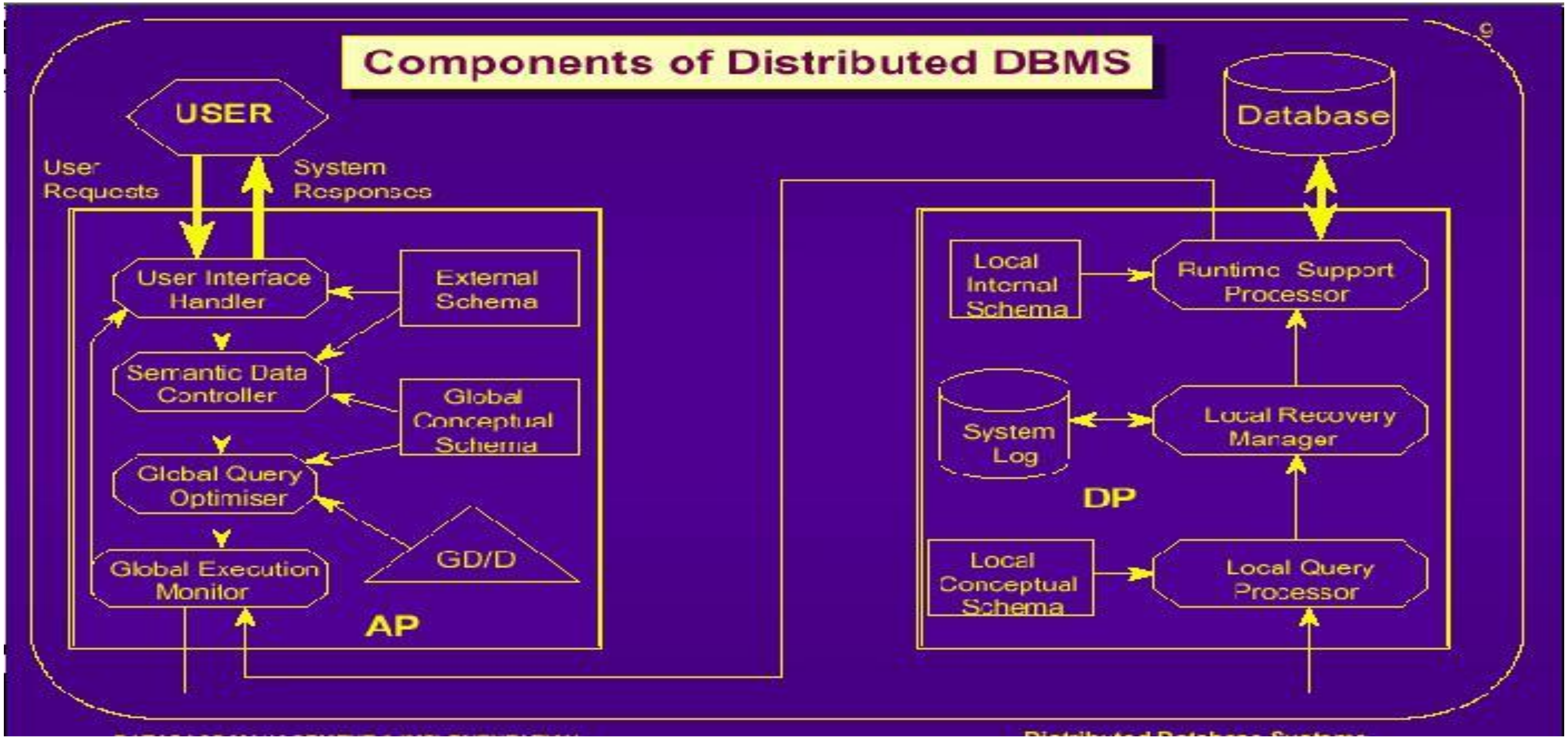
Peer-to-Peer Systems

- In peer-to-peer architectures there are no differentiations between the functionality of each site in the system[4].
- The physical data organization on each machine may be, and probably is, different.
- There needs to be an individual internal schema definition at each site, which we call the local internal schema (LIS).
- The enterprise view of the data is described by the global conceptual schema (GCS),
- To handle data fragmentation and replication, the logical organization of data at each site needs to be described.
- There needs to be a third layer in the architecture, the local conceptual schema (LCS).
- User applications and user access to the database is supported by external schemas (ESs), defined as being above the global conceptual schema.
- Location and replication transparencies are supported by the definition of the local and global conceptual schemas and the mapping in between them
- Network transparency, is supported by the definition of the global conceptual schema

Distributed Database Reference Architecture

Location and replication transparencies are supported by the definition of the local and global conceptual schemas and the mapping in between.




Components of Distributed DBMS
USER
User Requests
System Responses
User Interface Handler
External Schema
Semantic Data Controller
Global Conceptual Schema
Global Query Optimiser
GD/D
Global Execution Monitor
AP
Database
Local Internal Schema
Runtime Support Processor
System Log
Local Recovery Manager
DP
Local Conceptual Schema
Local Query Processor

Peer-to-Peer Systems

- User queries data irrespective of its location or of which local component of the distributed database system will service it.
- The 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.
- One component (**user processor**) handles the interaction with users, and another(**data processor**) deals with the storage

Peer-to-Peer Systems

- **Elements: Of User Processor(AP):**
  1. **User interface handler** is responsible for interpreting user commands as they come in, and formatting the result data as it is sent to the user.
  2. **Semantic data controller** uses the integrity constraints and authorizations that are defined as part of the global conceptual schema to check if the user query can be processed.
  3. **Global query optimizer and decomposer** determines an execution strategy to minimize a cost function, and translates the global queries into local ones using the global and local conceptual schemas as well as the global directory.
  4. **Global query optimizer** generates the best strategy to execute distributed join operations.
  5. **Distributed execution monitor/ distributed transaction manager** coordinates the distributed execution of the user request.

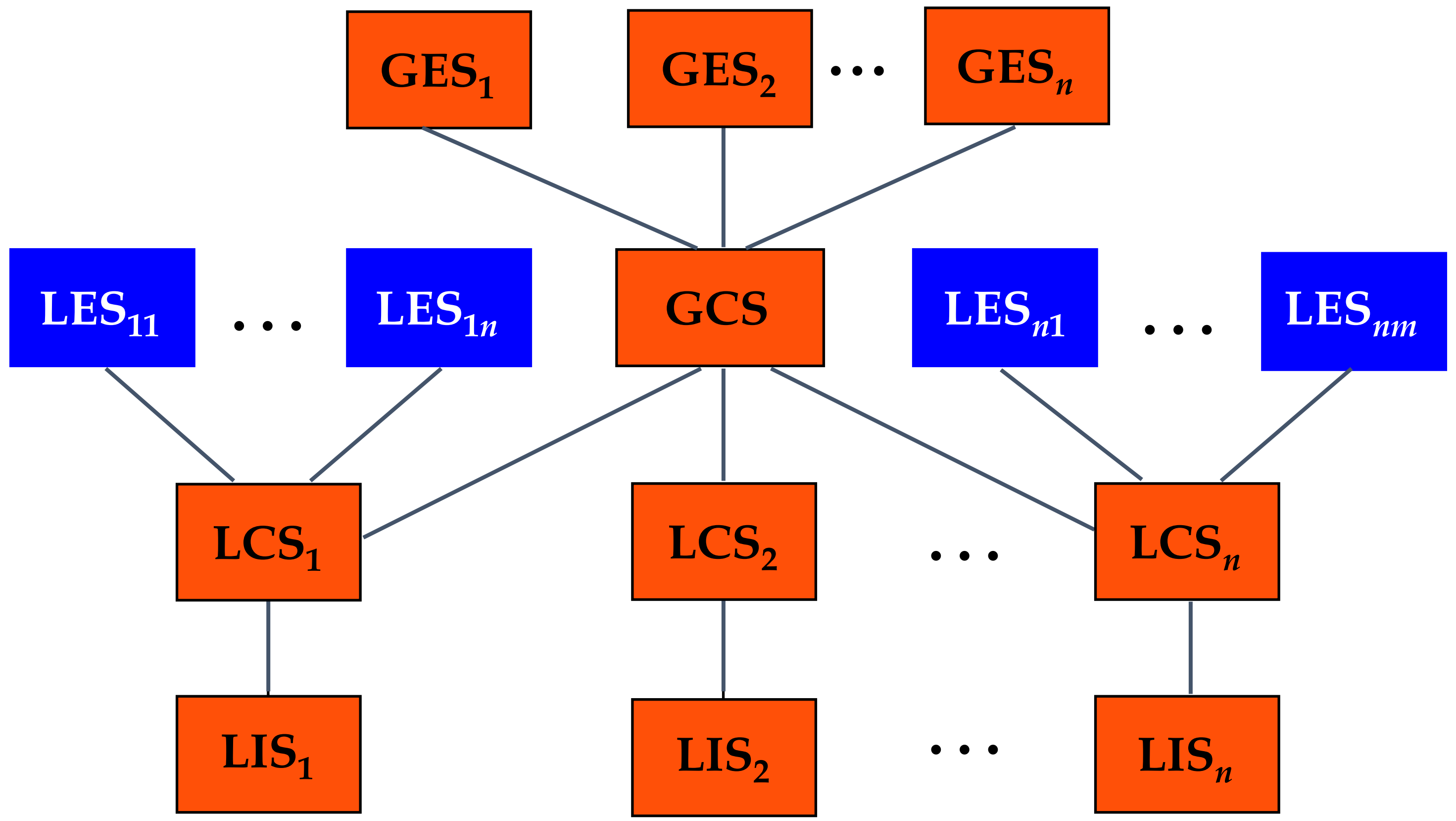
Peer-to-Peer Systems

- **Elements: Of Data Processor(DP):**
  1. **The local query optimizer,** acts as the access path selector, responsible for choosing the best access path to access any data item
  2. **The local recovery manager** is responsible for making sure that the local database remains consistent even when failures occur
  3. **The run-time support processor** physically accesses the database according to the physical commands in the schedule generated by the query optimizer.
     1. The run-time support processor is the interface to the operating system and contains the database buffer (or cache) manager, which is responsible for maintaining the main memory buffers and managing the data accesses.

MULTI-DATABASE SYSTEM ARCHITECTURE

- **Multidatabase systems (MDBS)** represent the case where individual DBMSs (whether distributed or not) are fully autonomous and have no concept of cooperation.
- they may not even "know" of each other's existence or how to talk to each other.
- In the case of logically integrated distributed DBMSs, the global conceptual schema (GCS) defines the conceptual view of the entire database, while in the case of distributed multi-DBMSs, it represents only the collection of some of the local databases that each local DBMS wants to share.
- In a MDBS, the GCS (which is also called a *mediated schema*) is defined by integrating either the external schemas of local autonomous databases or (possibly parts of their) local conceptual schemas.

DATALOGICAL MULTI-DBMS ARCHITECTURE



REVISION QUESTIONS

1. Draw a diagram and explain the levels of the ANSI/SPARC Architecture
2. Describe the six functions performed by each layer of the generic DBMS architecture.
3. How do these differ between Client/Server and Peer-to-Peer architectures?
4. Architectural models for distributed DBMSs can be characterised in three dimensions. Name and explain those dimensions
5. Compare client/server, peer-to-peer and multidatabase systems (MDBS) architectures in terms of these three dimensions.
6. Describe, with the aid of a diagram, the three-tier client/server distributed system architecture.
7. Describe purpose of the following components of the datalogical distributed DBMS architecture.
    - Local Internal Schema (LIS)
    - Global Conceptual Schema (GCS)
    - Local Conceptual Schema (LCS)
    - External Schemas (ES)
8. How are the various transparencies achieved in a Distributed Database Reference Architecture?