Distributed Database Management Systems

Chapter 2

Distributed DBMS Architecture

Contents

- Distributed DBMS Architecture
- ANSI/SPARC architecture:
- Centralized DBMS Architecture
- Architectural Models for DDBMSs
    - ✓Autonomy
    - ✓Distribution
    - ✓Heterogeneity
- Architectural Alternatives
    - ✓Client/server distributed DBMS
    - ✓Peer-to-peer distributed DBMS
    - ✓Heterogeneous multidatabase system

Distributed DBMS Architecture

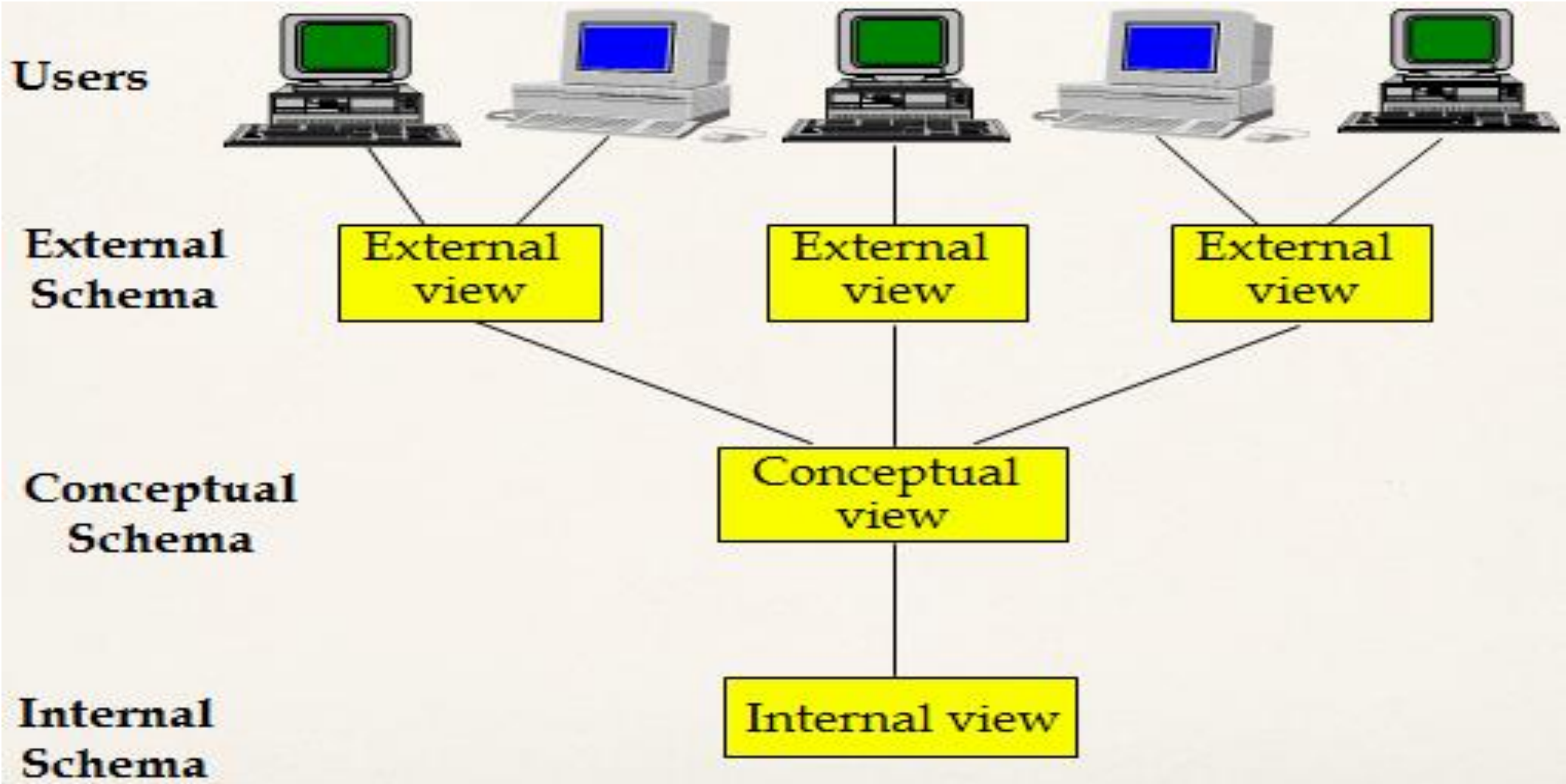
- Architecture of a system defines the structure of the system
  - → components identified
  - → functions of each component defined
  - → interrelationships and interactions between components defined
- "ANSI/SPARC architecture" is a *data logical* approach to defining a DBMS architecture.
  - → focuses on the different user classes and roles on data.
- A centralized DBMS architecture that extends to identify the set of alternative architectures for a distributed DBMS.
- Three "reference" architectures for a distributed DBMS:
  - → client/server systems
  - → peer-to-peer distributed DBMS, and
  - → multidatabase systems
- ANSI stands for ***American National Standards Institute*** and SPARC stand for ***Standards Planning and Requirement Committee***.

ANSI/SPARC architecture:

- The interfaces of this architecture were proposed to be standardized.
- Defines a framework that contains 43 interfaces, 14 of which deal with physical storage subsystem
- 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.
- For each of these views, an appropriate schema definition is required.
- At the lowest level of the architecture is the internal view
  - deals with the physical definition and organization of data.

ANSI/SPARC architecture(cont….)

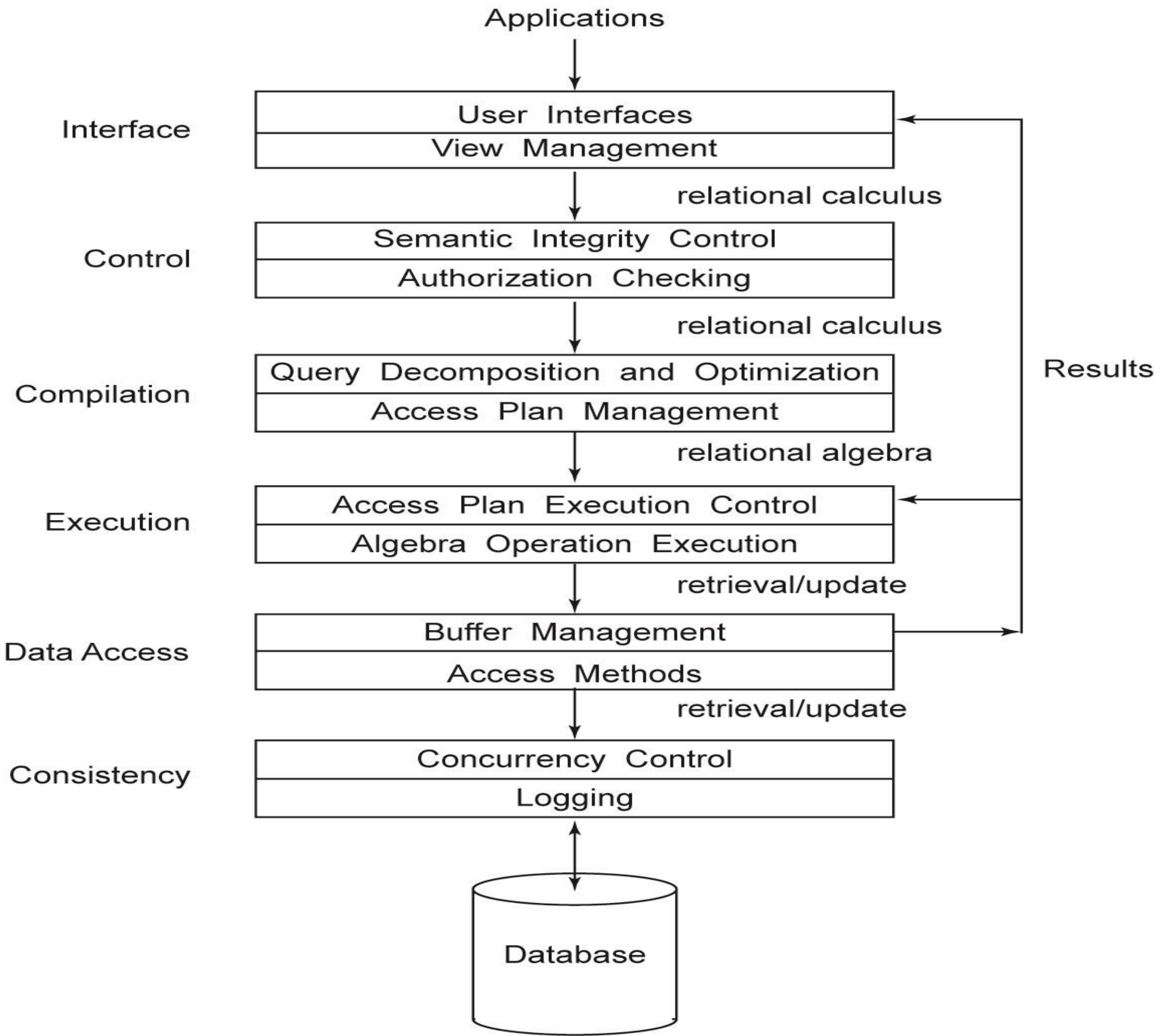
- At the other extreme is the external view, which is concerned with how users view the database.
- In between these two ends is the conceptual schema, which is an abstract definition of the database.
  - → It is the "real world" view of the enterprise being modeled in the database



Centralized DBMS Architecture

- A centralized DBMS is interfaced with two other components:
  - the communication subsystem - permits interfacing the DBMS with other subsystems in order to communicate with applications
  - the operating system - provides the interface between the DBMS and computer resources (processor, memory, disk drives, etc.).
- The functions performed by a DBMS can be layered: the layers are the
  - interface,
  - control,
  - compilation,
  - execution,
  - data access, and
  - consistency management

Centralized DBMS Architecture(cont…)

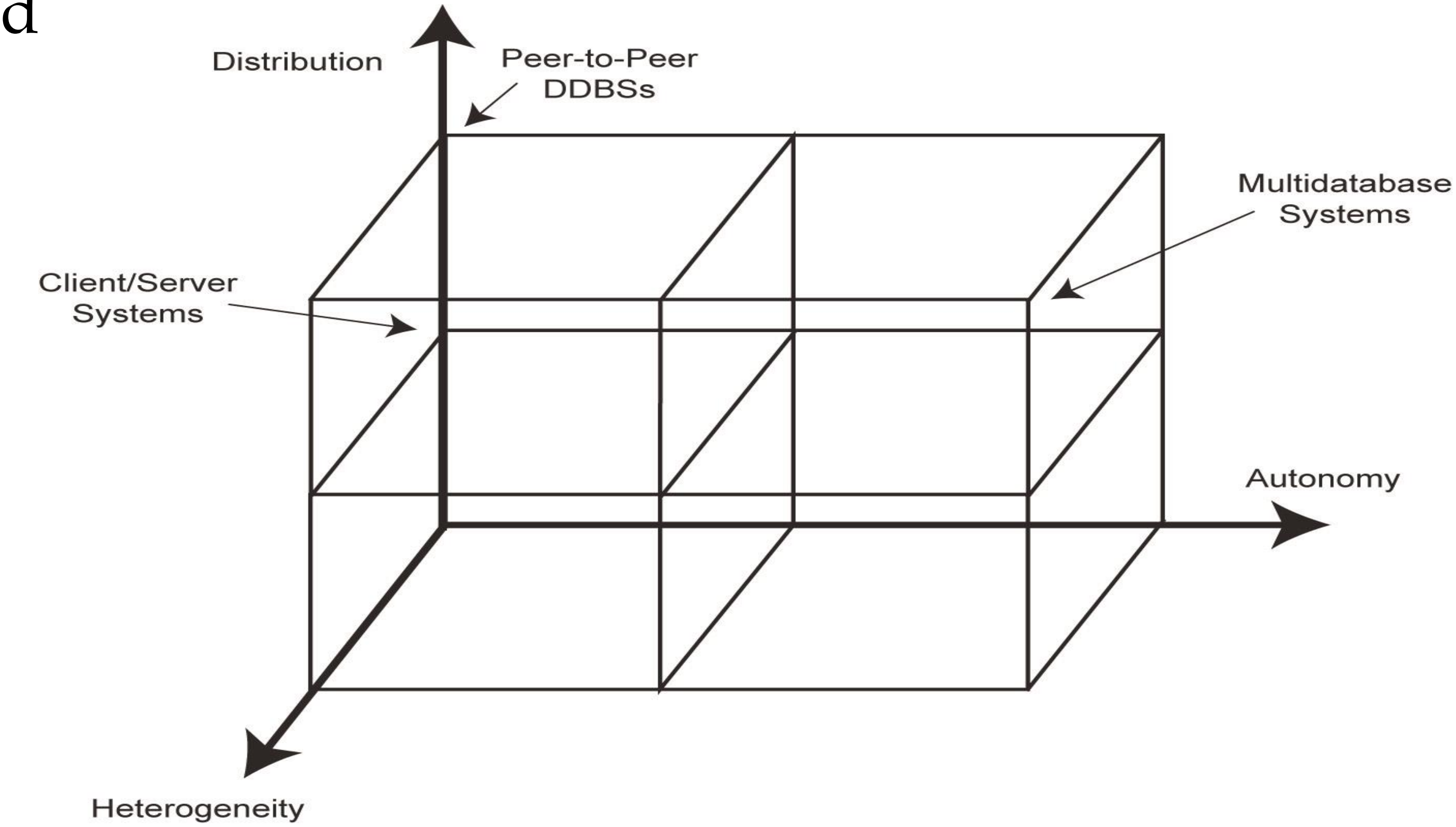


Centralized DBMS Architecture(cont…)

- The interface layer manages the interface to the applications.
- The control layer controls the query by adding semantic integrity predicates and authorization predicates.
- The query processing (or compilation) layer maps the query into an optimized sequence of lower-level operations.
- The execution layer directs the execution of the access plans, including transaction management and synchronization of algebra operations
- The data access layer manages the data structures that implement the files, indices, etc.
- The consistency layer manages concurrency control and logging for update requests.

Architectural Models for DDBMSs

- Architectural models for DDBMS can be characterized with respect to three different dimensions:
  - → the autonomy of local systems,
  - → their distribution, and
  - → their heterogeneity.



Architectural Models for DDBMSs(Autonomy)

- Autonomy refers to the distribution of control, not of data.
  - It indicates the degree to which individual DBMSs can operate independently.
- Dimensions of autonomy can be specified as:
  - Design autonomy: Ability of a component DBMS to decide on issues related to its own design.
  - Communication autonomy: Ability of a component DBMS to decide whether and how to communicate with other DBMSs.
  - Execution autonomy: Ability of a component DBMS to execute local operations in any manner it wants to.

Architectural Models for DDBMSs(Autonomy) cont..

- Dimensions of autonomy are covered in three different aspects:
  - Tight integration: a single-image of the entire database is available to any user
  - Semiautonomous systems: Consist of DBMSs that can operate independently and determine what parts of their own database will be accessible to users of other DBMSs.
  - Total isolation: Individual systems are stand-alone DBMSs that know neither of the existence of other DBMSs nor how to communicate with them.

Architectural Models for DDBMSs(Distribution)

- The distribution dimension of the taxonomy deals with the physical distribution of data over multiple sites.
- Two classes of distribution are:
  - Client/server distribution: Concentrâtes data management duties at servers while the clients focus on providing the application environment including the user interface.
  - Peer-to-peer distribution (or full distribution): There is no distinction of client machines versus servers.
    - Each machine has full DBMS functionality and can communicate with other machines to execute queries and transactions

Architectural Models for DDBMSs(Heterogeneity)

- Heterogeneity may occur in various forms in DDBS:
  - → Heterogeneity in data models - Created by the data representation with different modeling tools.
  - → Heterogeneity in query languages - involves differences in languages.
  - → Heterogeneity in transaction management protocols - the use of different data access paradigms in different data models.

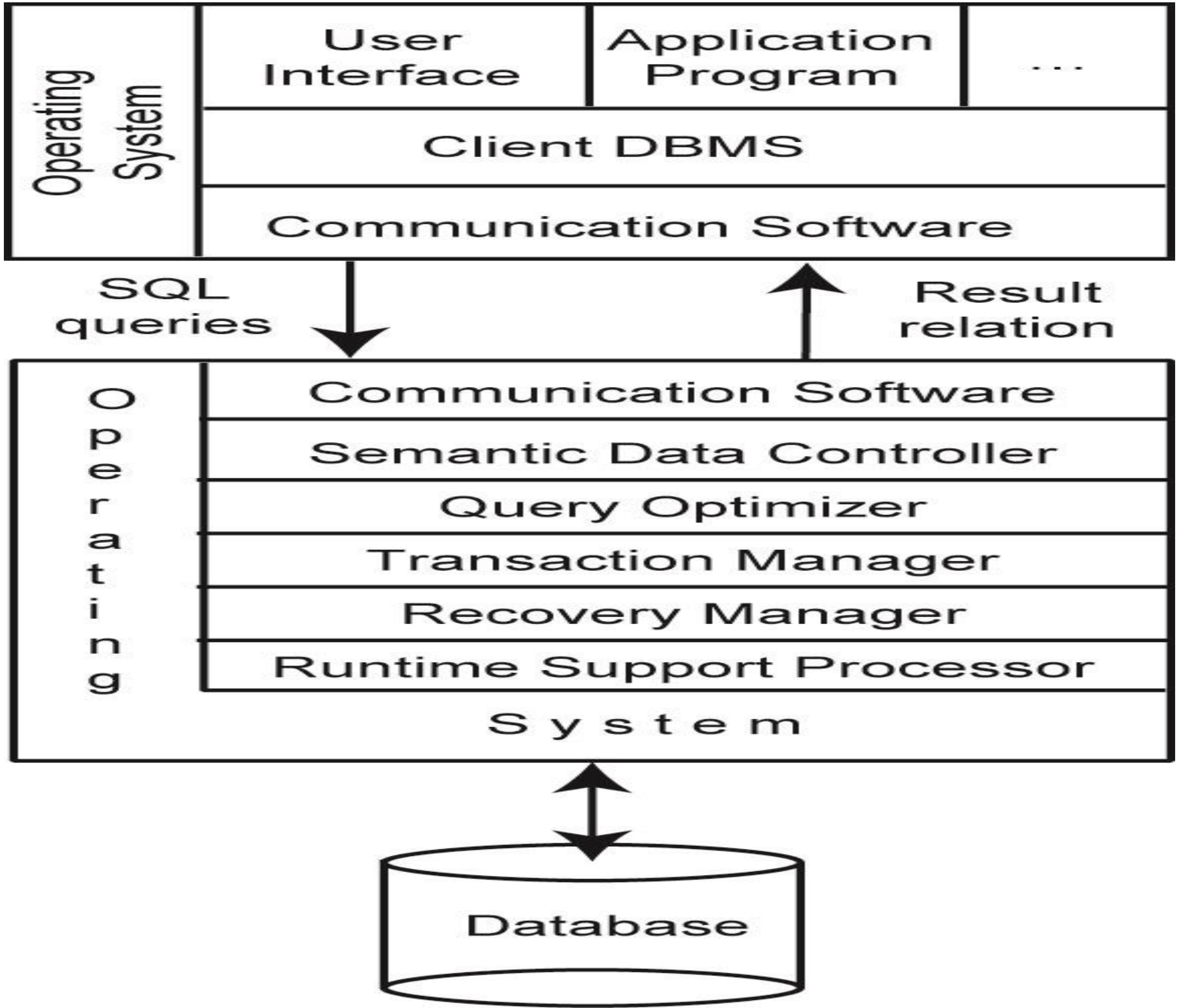
Architectural Alternatives

- Three alternative architectures are:
  - Client/server distributed DBMS – provides a two-level architecture based on server functions and client functions
  - Peer-to-peer distributed DBMS - no differentiation between the functionality of each site in the system.
  - Heterogeneous multidatabase system - individual DBMSs (whether distributed or not) are fully autonomous and have no concept of cooperation.

Architectural Alternatives (Client/server distributed DBMS)

- Provides a two-level architecture based on server functions and client functions.
- Server stores and administers the data.
- All of query processing and optimization, transaction management and storage management is done at the server.
- The client provides:
  - the application and the user interface
  - a DBMS client module - responsible for managing the data that is cached to the client
  - The management of transaction lock that is cached to the client
  - consistency checking of user queries at the client side
- Communication between the clients and the server(s) is at the level of SQL statements.
  - Client passes SQL queries to the server, the server returns the result

Architectural Alternatives (Client/server distributed DBMS) cont….

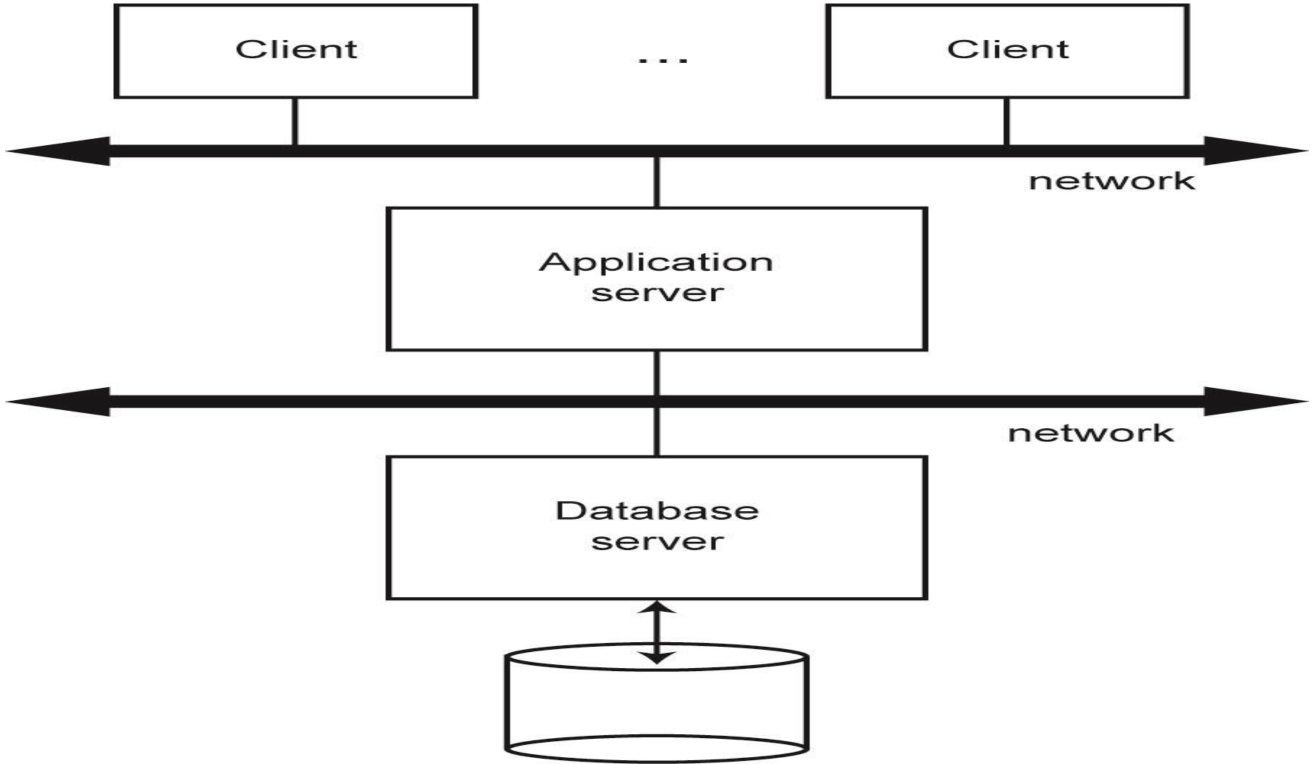


Architectural Alternatives (Client/server distributed DBMS) cont….

- **Two simple types of client/server architecture:**
  - multiple client/single server : the database is stored only on the server that also hosts the software to manage it.
  - multiple client/multiple server :
    - two alternative management strategies:
      - ✓ either each client manages its own connection to the appropriate server or
      - ✓ each client knows of only its “home server” which then communicates with other servers as required

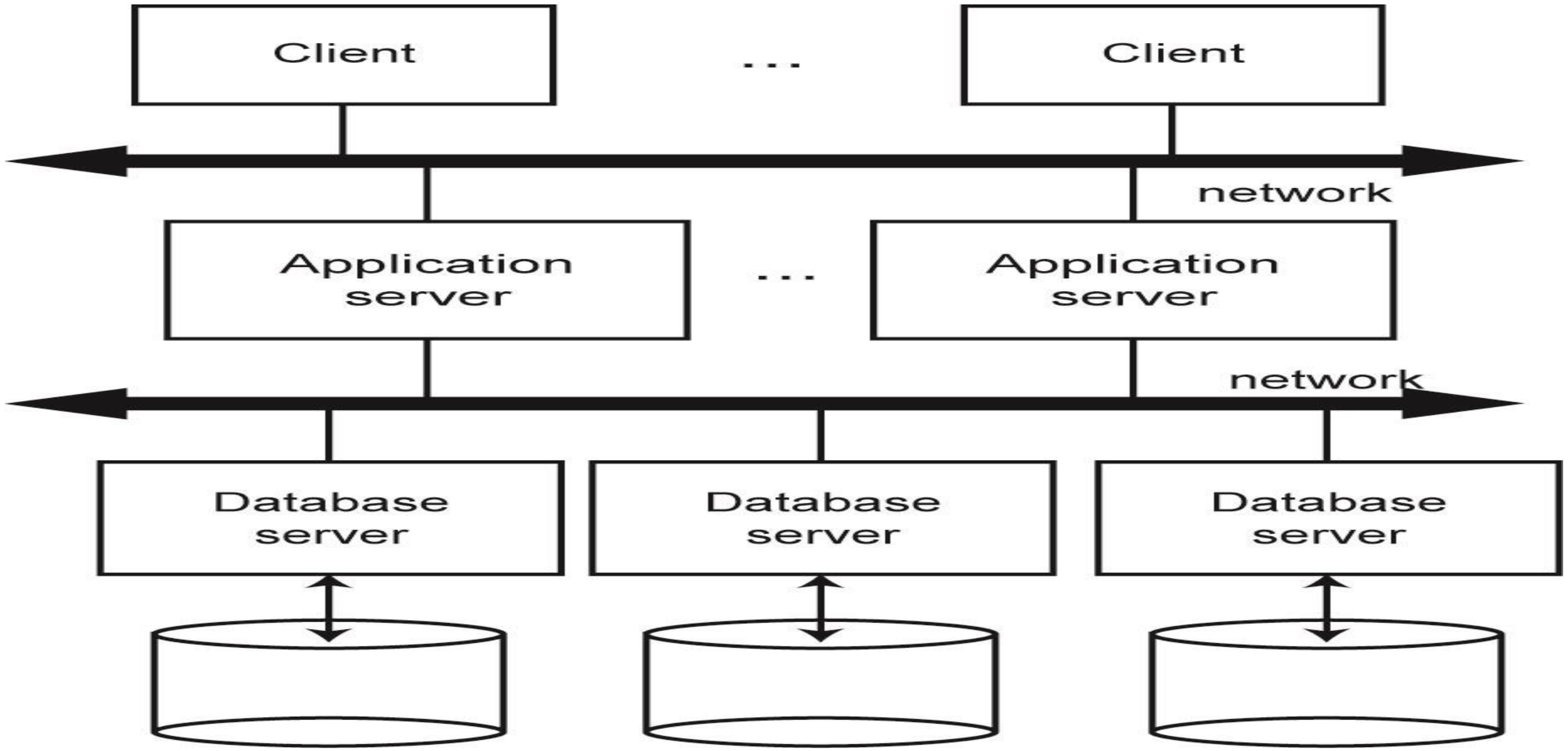
Architectural Alternatives (Client/server distributed DBMS) cont.…

- **Three types of servers in client/server architecture:**
  - client servers run the user interface (e.g., web servers),
  - application servers run application programs, and
  - database servers run database management functions.
- **Data Base Server Approach:** An extension of the client/server architecture, with application servers connected to one database server via a communication network.



Architectural Alternatives (Client/server distributed DBMS) cont.…

- **Distributed Data Base Servers Approach:**
  - Introduces 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

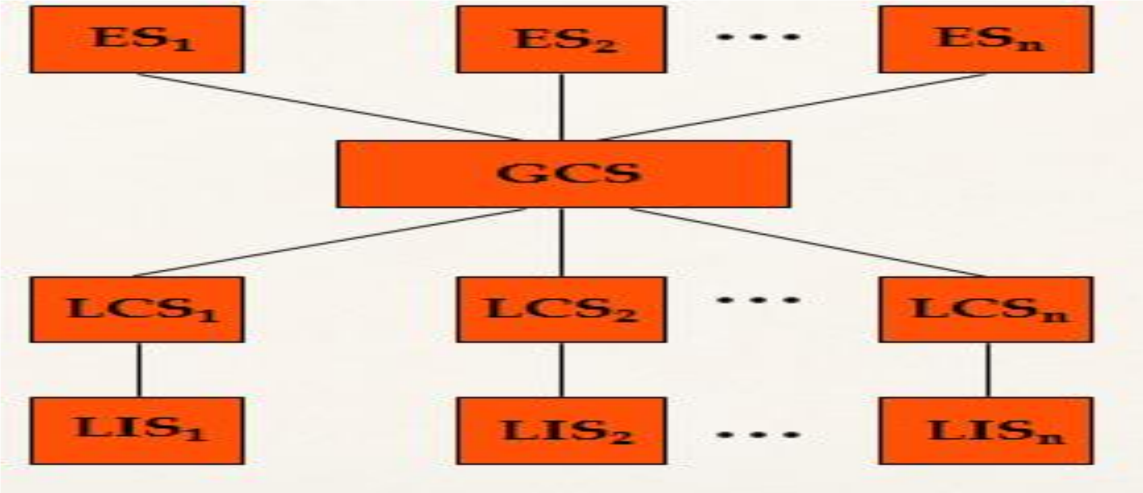


Architectural Alternatives (Peer-to-peer distributed DBMS)

- In peer-to-peer architecture:
  - there is no differentiation between the functionality of each site in the system.
  - there is inherent heterogeneity of every aspect of the sites and their autonomy
  - the physical data organization on each machine may be different:
    - local internal schema (LIS) - an individual internal schema definition at each site.
    - global conceptual schema (GCS) - The enterprise view of the data, which is global and describes the logical structure of the data at all the sites.
    - local conceptual schema (LCS) - handles data fragmentation and replication at each site
    - external schemas (ESs) – supports user applications and user access to the database

Architectural Alternatives (Peer-to-peer distributed DBMS)cont.…

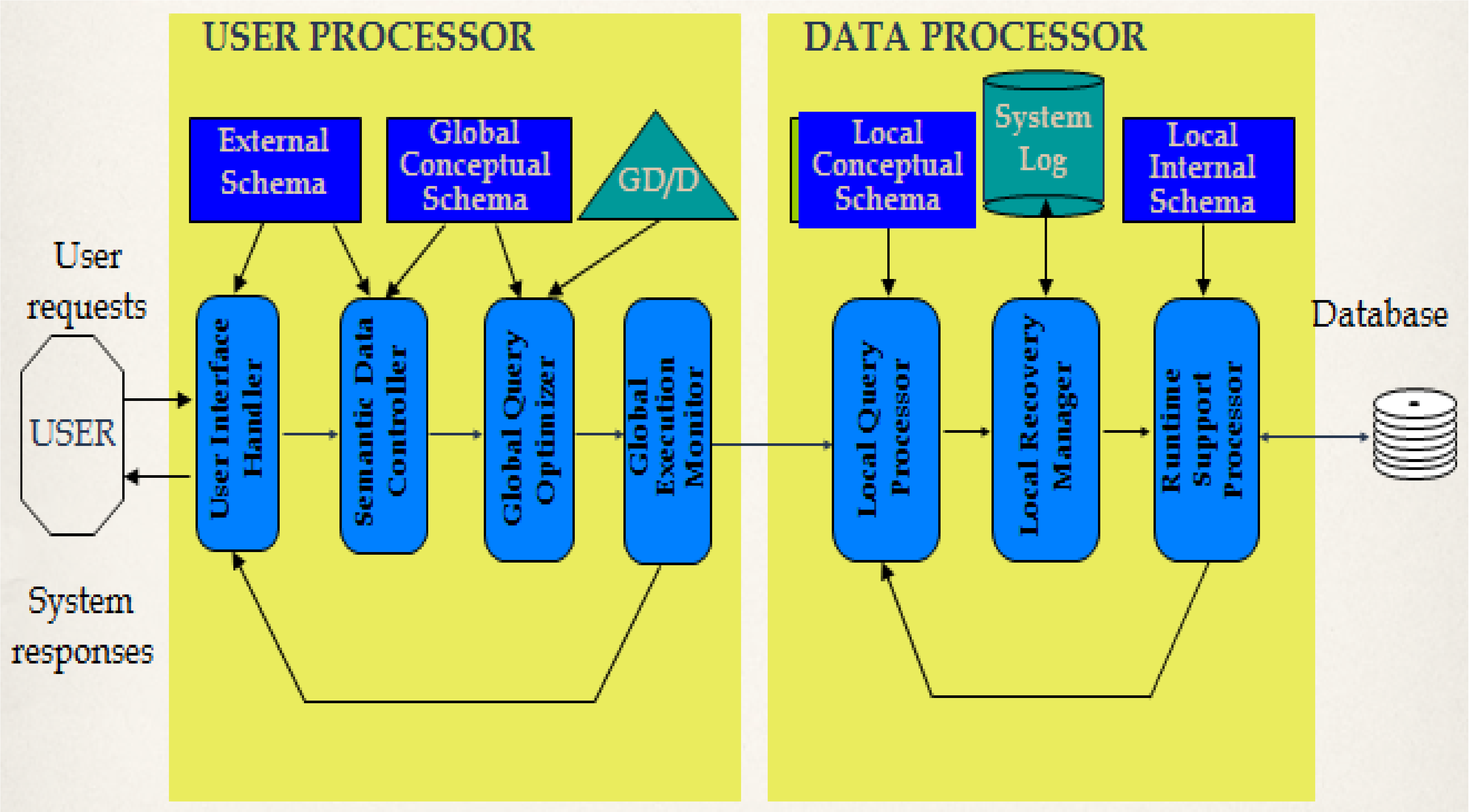
Peer-to-peer Datalogical DDBS Architecture:



- Data independence is supported as the model is an extension of ANSI/SPARC
- Location and replication transparencies are supported by the definition of LCSs and GCSs schemas and the mapping in between.
- Network transparency is supported by the definition of the GCS
- The DDBS translates global queries into a group of local queries, which are executed by DDBMS components at different sites that communicate with one another.

Architectural Alternatives (Peer-to-peer distributed DBMS)cont….

The detailed components of a distributed DDBMS are shown in the following Figure:



Architectural Alternatives (Peer-to-peer distributed DBMS)cont.…

- Two first major components of peer-to-peer DDBMS are:
  - User processor
  - Data processor
- User processor consists of four elements:
  - user interface handler - interprets user commands and formats the result data as it is sent to the user.
  - semantic data controller - uses the integrity constraints and authorizations (defined as part of GCS) to check if the user query can be processed.
  - global query optimizer and decomposer - determines an execution strategy to minimize a cost function, and translates the global queries into local ones using GCS, LCS and global directory
  - distributed execution monitor (or distributed transaction manager)-coordinates distributed execution of the user request.

Architectural Alternatives (Peer-to-peer distributed DBMS)cont.…

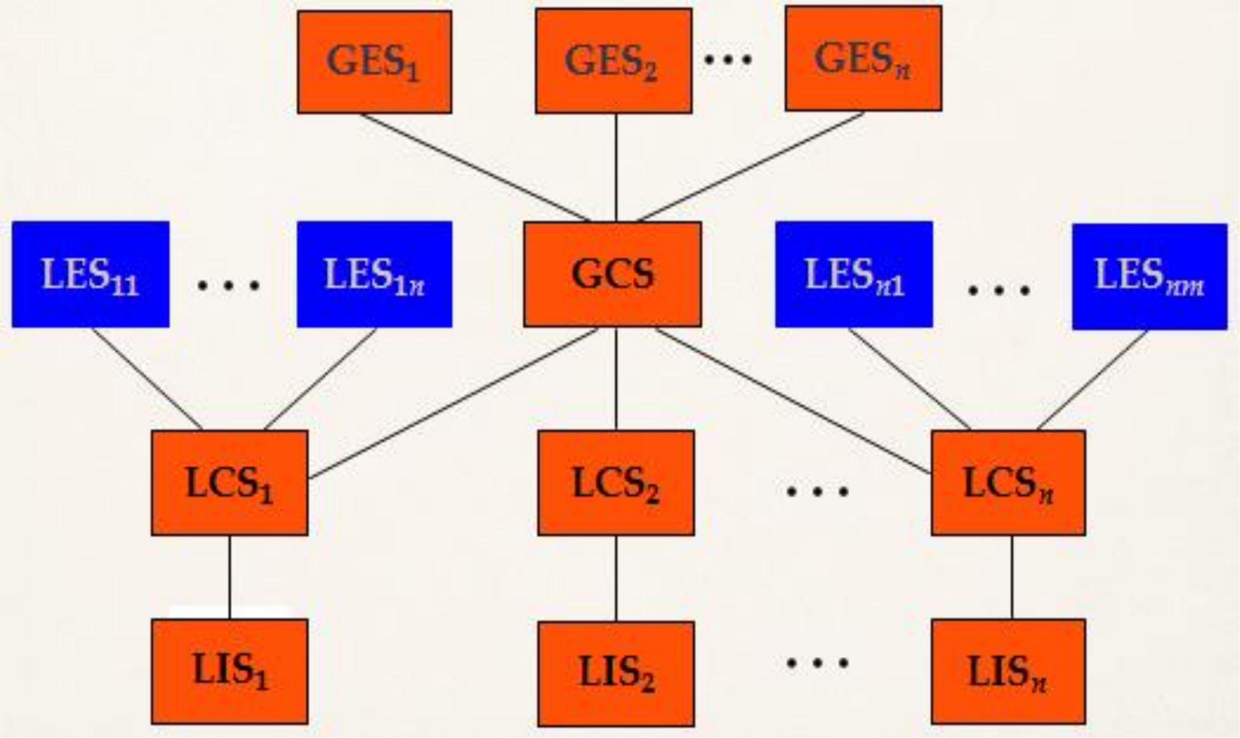
- Data processor consists of three elements:
  - → local query optimizer - acts as the access path selector responsible for choosing the best access path to access any data item
  - → local recovery manager - responsible for making sure that the local database remains consistent even when failures occur
  - → run-time support processor - physically accesses the database according to the physical commands in the schedule generated by the query optimizer

Architectural Alternatives (Multidatabase System Architecture)

- In Multidatabase systems (MDBS), the individual DBMSs (whether distributed or not) are fully autonomous and have no concept of cooperation.
- **Distributed DBMS vs. Distributed Multi-DBMS:**
  - In distributed DBMS, the GCS defines the conceptual view of the entire database;
    - in distributed MDBMS, it represents only the collection of some of the local databases that each local DBMS wants to share.
  - Global database in the latter is equal to the union of local databases,
    - whereas in the former it is only a (possibly proper) subset of the same union.

Architectural Alternatives (Multidatabase System Architecture) cont…

- **Distributed DBMS vs. Distributed Multi-DBMS:**
  - ➞ Difference between the design of the GCS in multi-DBMSs and distributed DBMSs:
    - ✦ in the former the mapping is from LCS to a GCS. In the latter, however, mapping is in the reverse direction.
- Datalogical Multi-DBMS Architecture :



Architectural Alternatives (Multidatabase System Architecture) cont….

- If heterogeneity exists in the system, then two implementation alternatives exist:
  - unilingual MDBMS and
  - Multilingual MDBMS