Lecture 2

Introduction to distributed databases system

Generic DBMS Architecture

- The functions performed by a DBMS can be layered .
- Taking a top-down approach, the layers are the **interface**,

control, **compilation** , **execution**, **data access**, and **consistency management**.

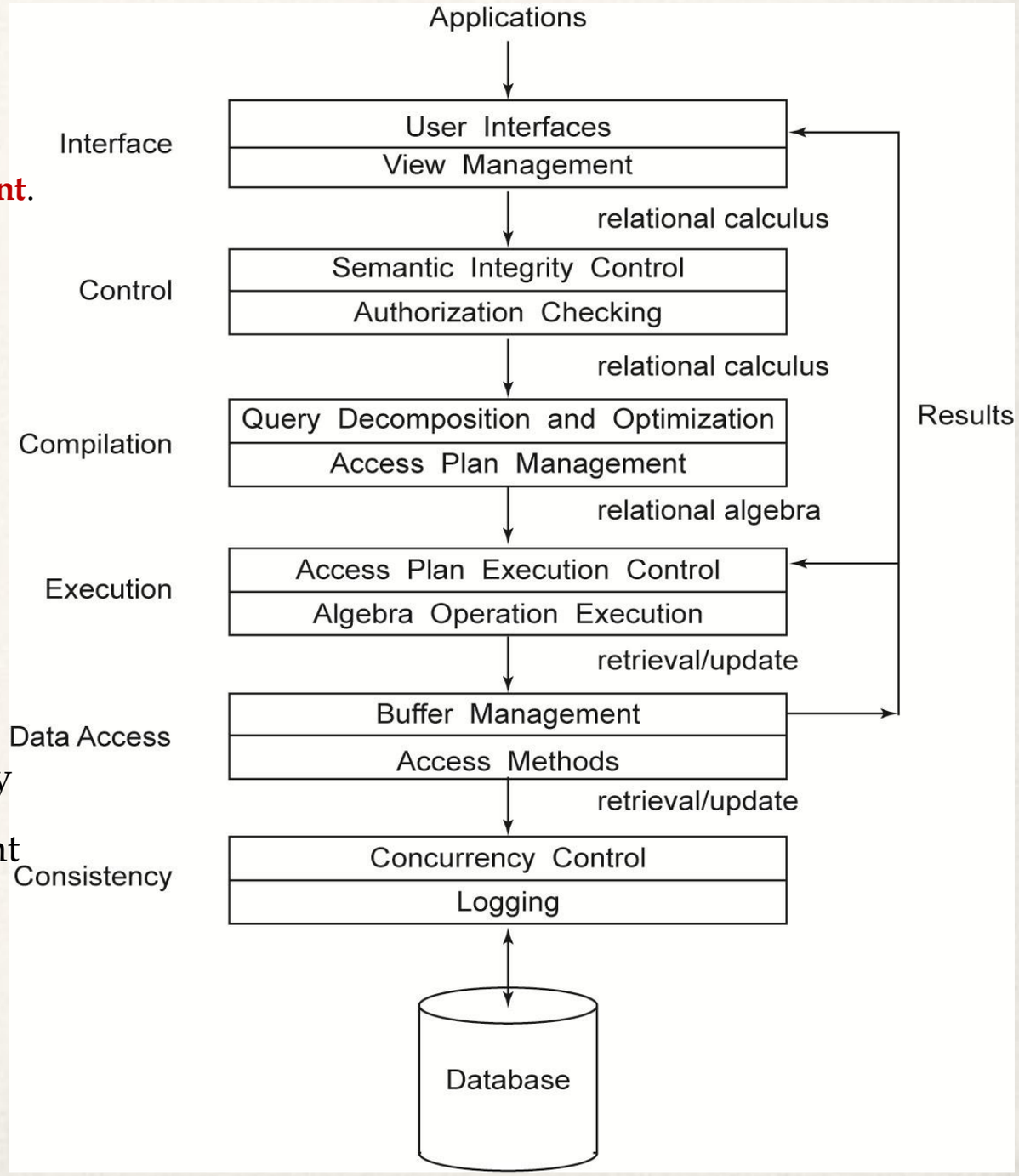
1. The interface layer

- manages the interface to the applications.

E.g. SQL statement in RDBMS.

2. The control layer

- **semantic integrity control:**
  - checks integrity constraints as unique key constraints, and Foreign key constraint.
  - Semantic integrity control ensures database consistency by rejecting update transactions that lead to inconsistent database states
- **Authorization Checking**
  - Check whether the **user** can execute the **operation** on the object.



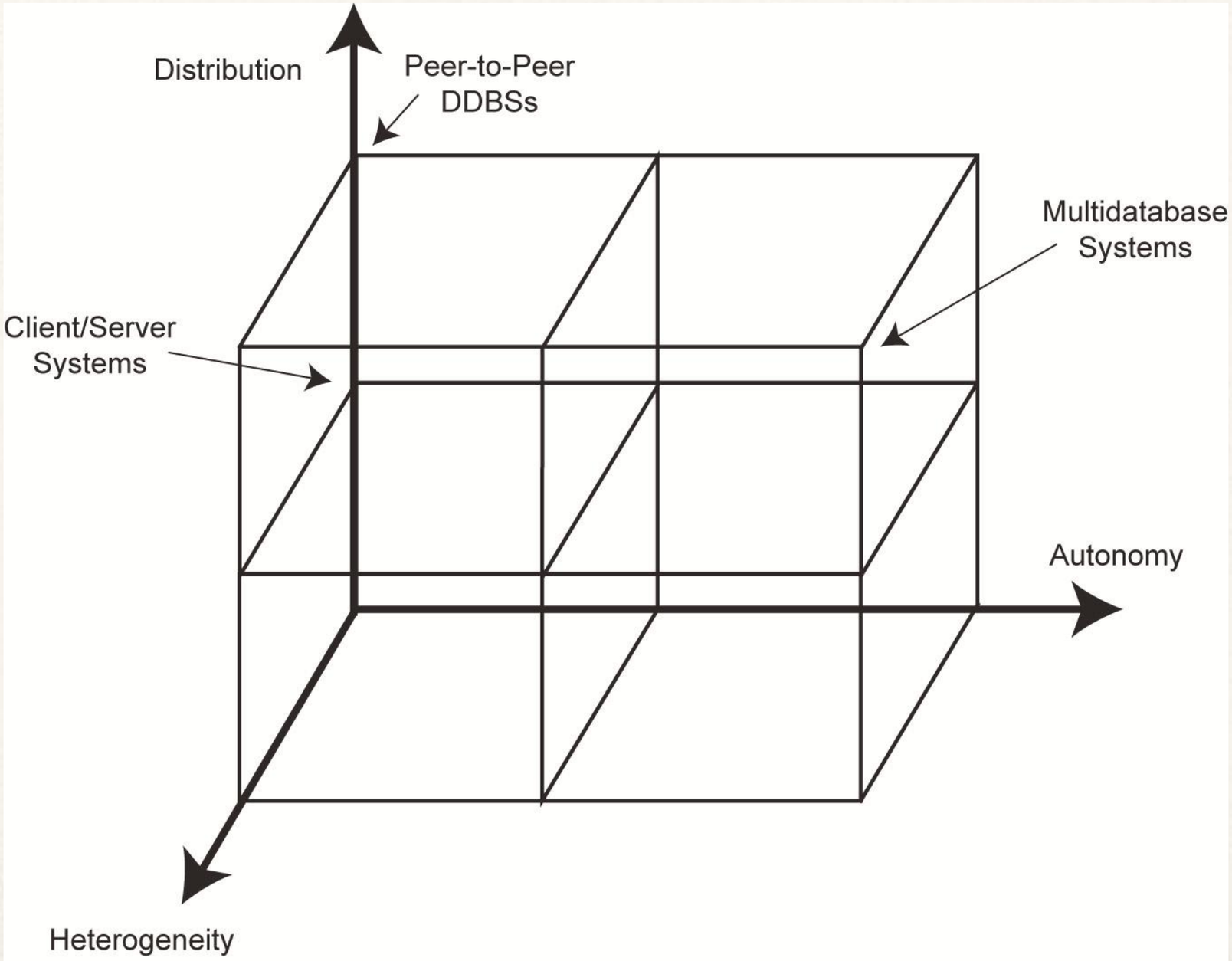
Generic DBMS Architecture*(cont)*

3. The query processing (or compilation) layer

- maps the query into an optimized sequence of lower-level operations.
- The output of this layer is a query expressed in lower-level code (algebra operations).
- This layer is concerned with performance.

- **The execution layer**
  - directs the execution of the access plans, including transaction management (commit, restart) and synchronization of algebra operations.
  - It interprets the relational operations by calling the data access layer through the retrieval and update requests.
- **The data access layer:**
  - manages the data structures that implement the files, indices, etc.
  - It also manages the buffers by caching the most frequently accessed data.
  - Careful use of this layer minimizes the access to disks to get or write data.
- **Finally, the consistency layer:**
  - manages concurrency control and logging for update requests.
  - This layer allows transaction, system, and media recovery after failure.

DBMS Implementation Alternatives



Dimensions of the Problem

- **Autonomy: Requirements of an autonomous system have been specified in D-DBMS as follows:**
  - The local operations of the individual DBMSs are not affected by their participation.
  - The manner in which the individual DBMSs process queries and optimize them should not be affected by the execution of global queries in the distributed system.
  - System consistency or operation should not be compromised when individual DBMSs join or leave the distributed system.

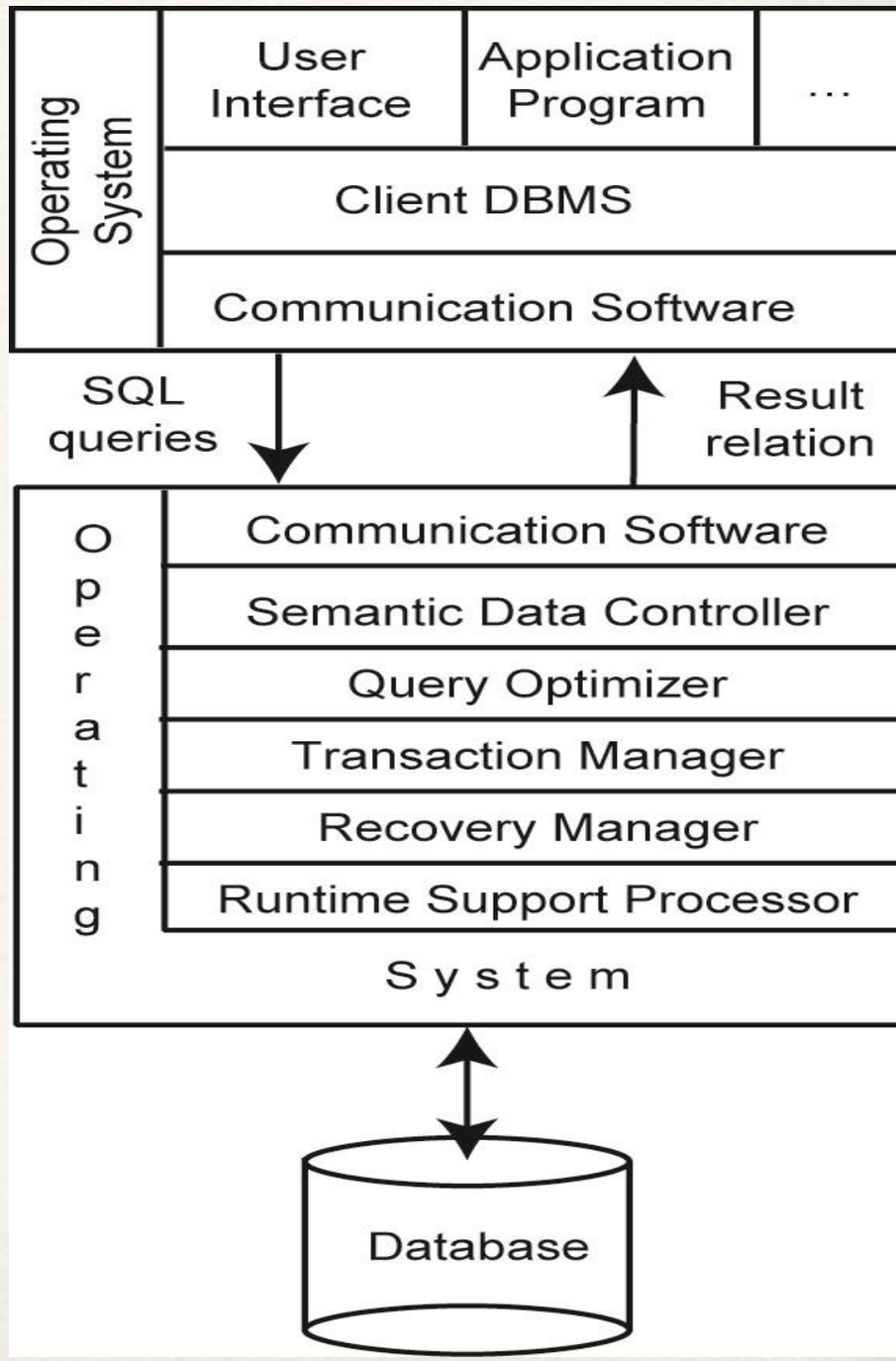
- **Various versions of autonomy**
  - **Design autonomy:** Ability of a component DBMS to decide on issues related to its own design.
  - **Communication autonomy**: Ability of a component DBMS to decide whether and how to communicate with other DBMSs.
  - **Execution autonomy**: Ability of a component DBMS to execute local operations in any manner it wants to.

Dimensions of the Problem

- **Distribution**
  - ➡ Whether the components of the system are located on the same machine or not and the physical distribution of data over multiple sites
    - ✦ The client/server distribution concentrates data management duties at servers while the clients focus on providing the application environment including the user interface.
    - ✦ In peer-to-peer systems, 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.
- **Heterogeneity**
  - ➡ Various levels (hardware, Networking protocols, operating system)
  - ➡ DBMS important one
    - ✦ data model, query language, transaction management algorithms

Client/Server Architecture

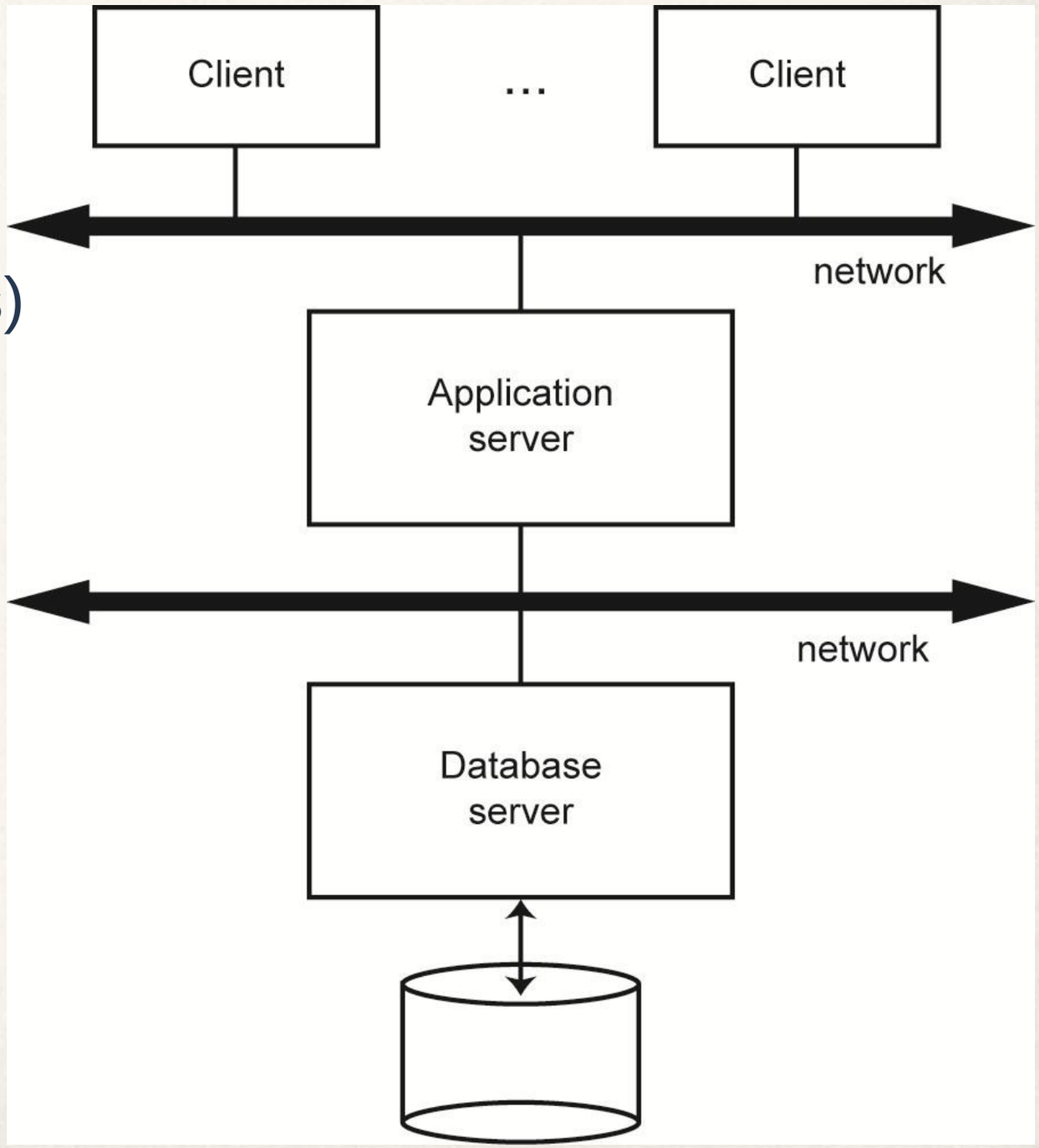
- the client passes SQL queries to the server without trying to understand or optimize them. The server does most of the work and returns the result relation to the client.



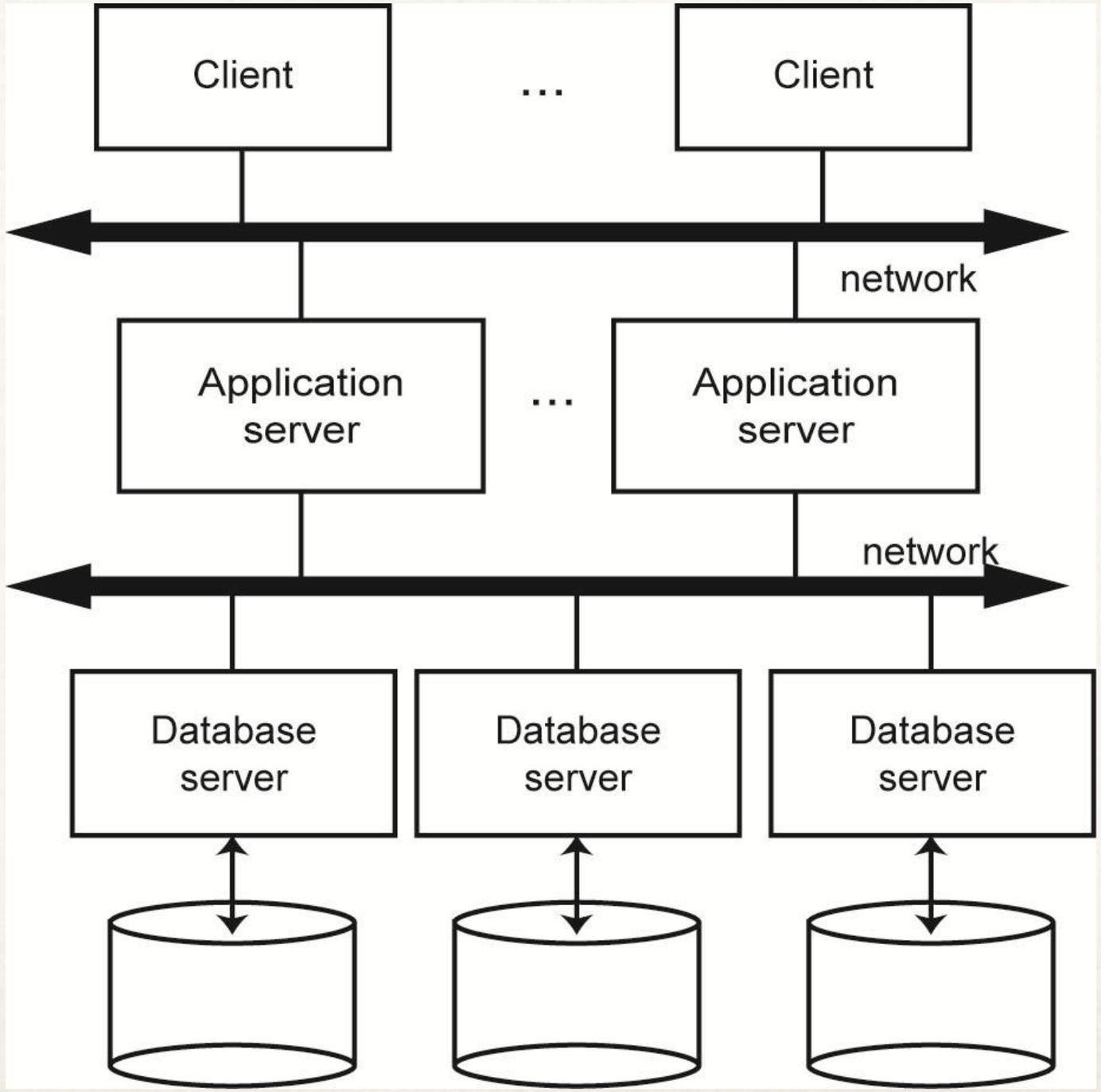
- DBMS client module is responsible for
  - ➡ managing the data that is cached to the client
  - ➡ managing the transaction locks

Multiple client/single server approach

Client servers
run the user interface(e.g. web servers)
Application servers
run application programs.
Database servers
run database management functions.

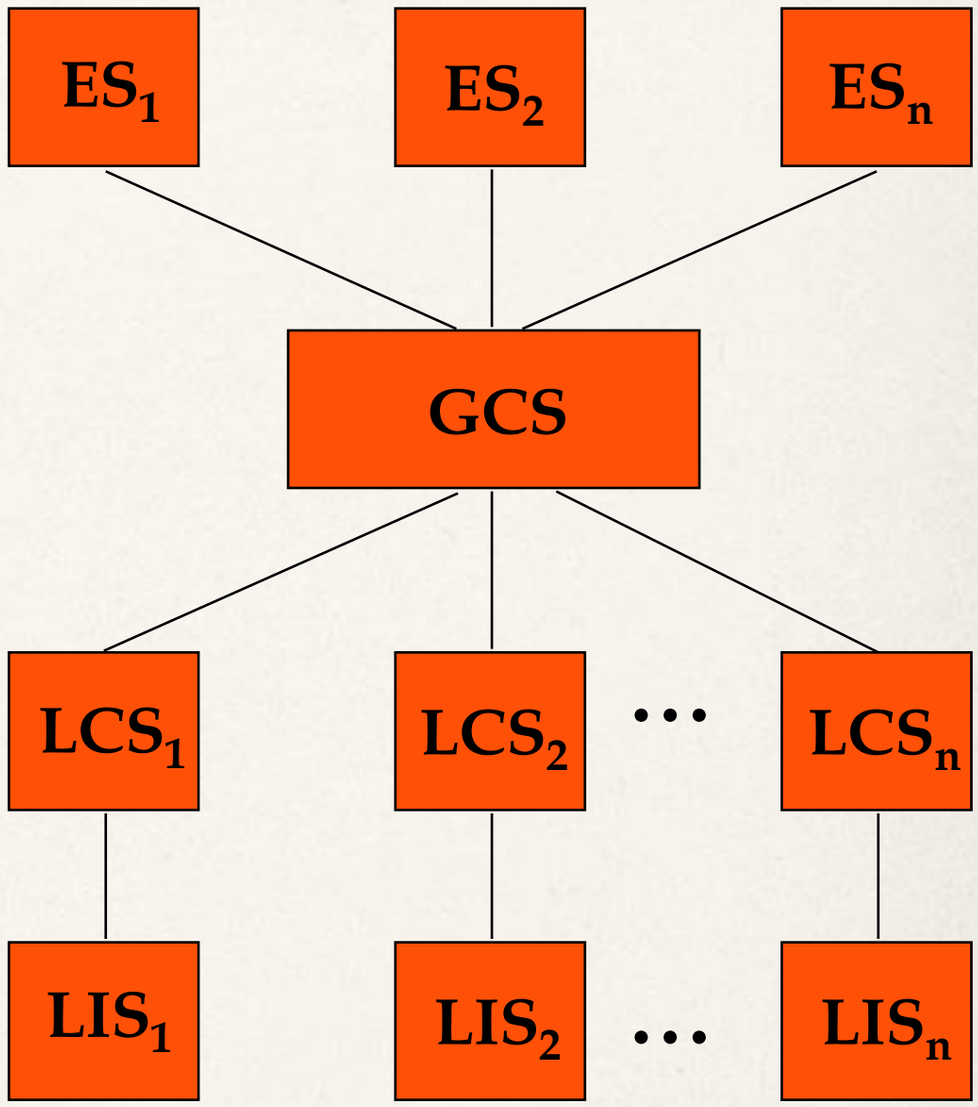


Multiple client/multiple server approach

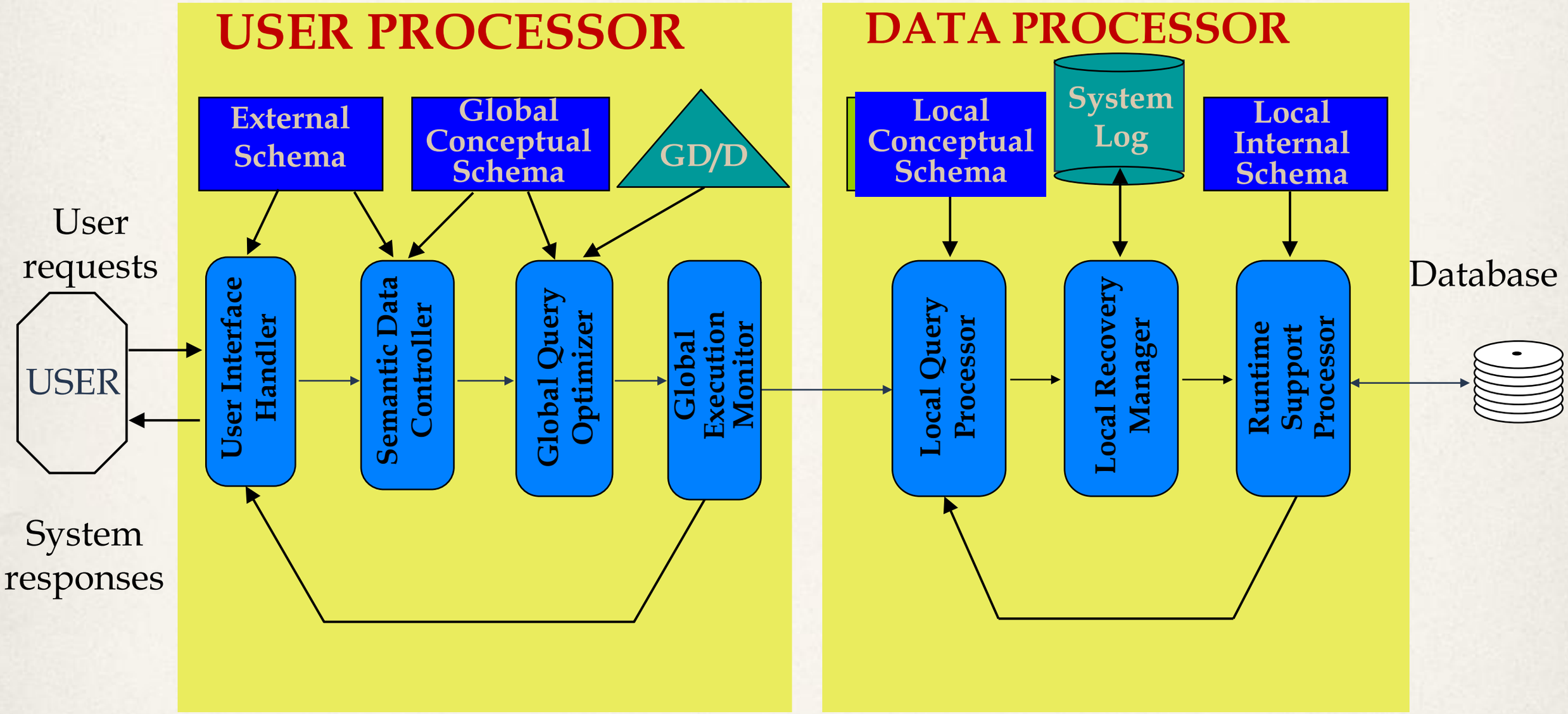


Distributed DBMS based peer- to peer Architecture

- **local internal schema (LIS).**
  - ➡ Internal schema definition at each site
- **conceptual schema (GCS).**
  - ➡ The enterprise view of the data.
- **local conceptual schema (LCS).**
  - ➡ handle data fragmentation and replication at each site
- **external schemas (ESs).**
  - ➡ Supports user applications and user access to the database



Peer-to-Peer Component Architecture



user processor consists of four elements

- **1.The 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. The 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. The global query optimizer and decomposer** determine the best strategy to execute distributed join operations to minimize a cost function .
- **4. The distributed execution monitor** coordinates the distributed execution of the user request. The execution monitor is also called the distributed transaction manager.

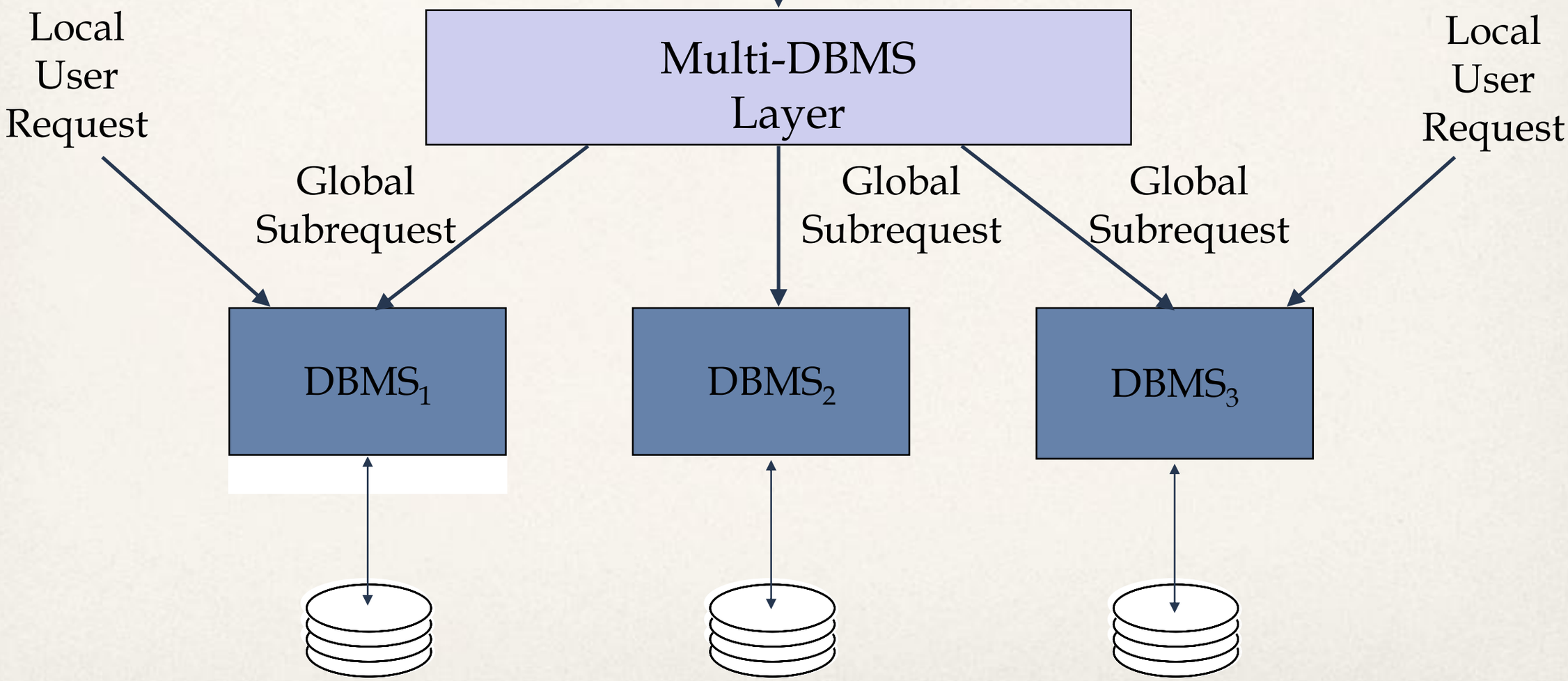
data processor and consists of three elements:

- **1. The local query optimizer**, acts as the access path selector, is 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.
  - ➡ 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.

MDBS Components & Execution

- Individual DBMSs are fully autonomous and have no concept of cooperation.

- May not even "know" of each other's existence or how to talk to each other.

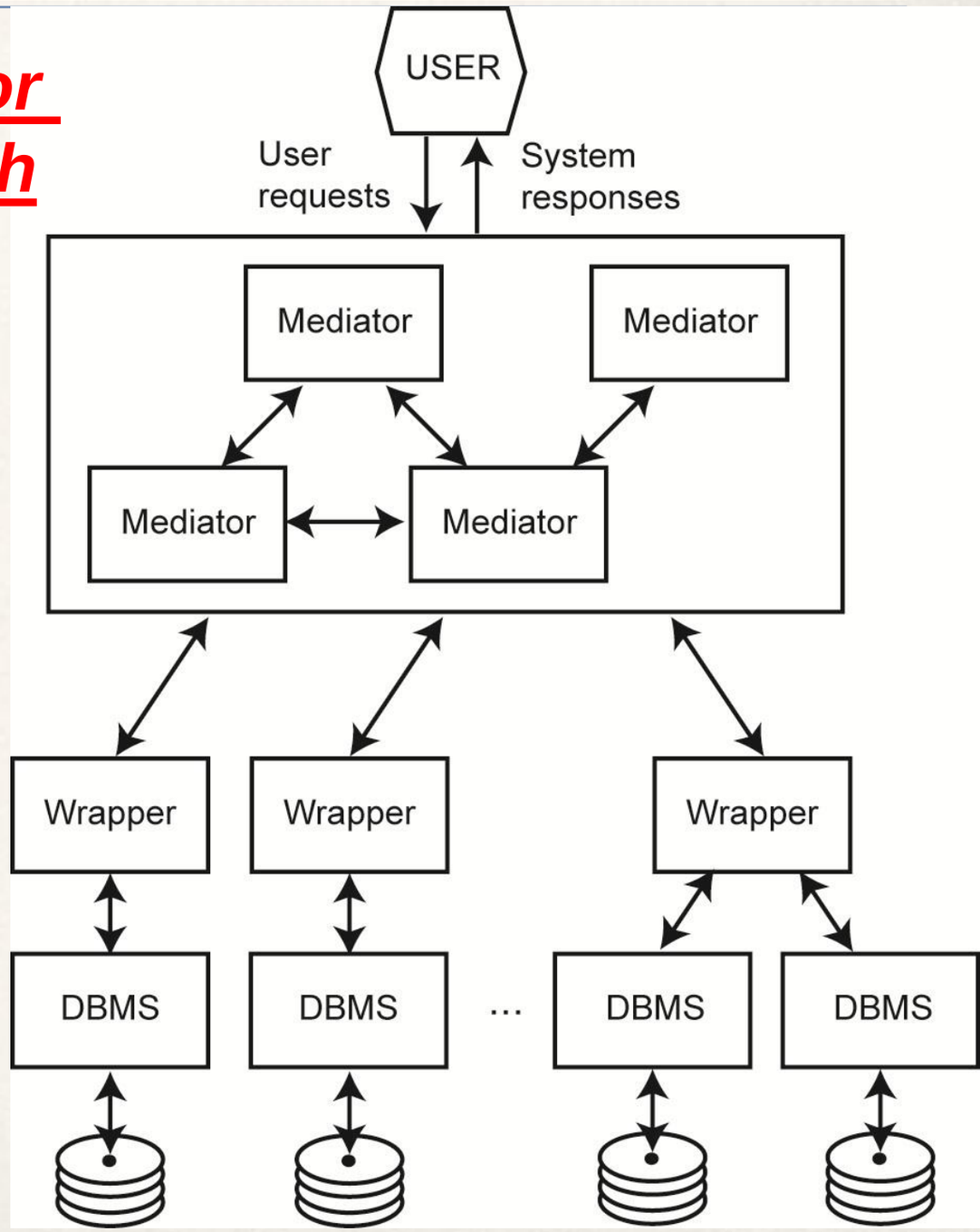


Mediator/Wrapper Architecture

A popular implementation architecture for MDBSs is the mediator/wrapper approach

A mediator "is a software module that exploits encoded knowledge about certain Sets data to create information for a higher layer of applications.

wrappers provide a mapping between a source DBMSs view and the mediators'
For example,
if the source DBMS is a relational one, but the mediator implementations are object-oriented, the required mappings are established by the wrappers.



Questions