Systems Infrastructure for Data Science

Web Science Group
Uni Freiburg

WS 2012/13

Lecture VII: Introduction to Distributed Databases

Why do we distribute?

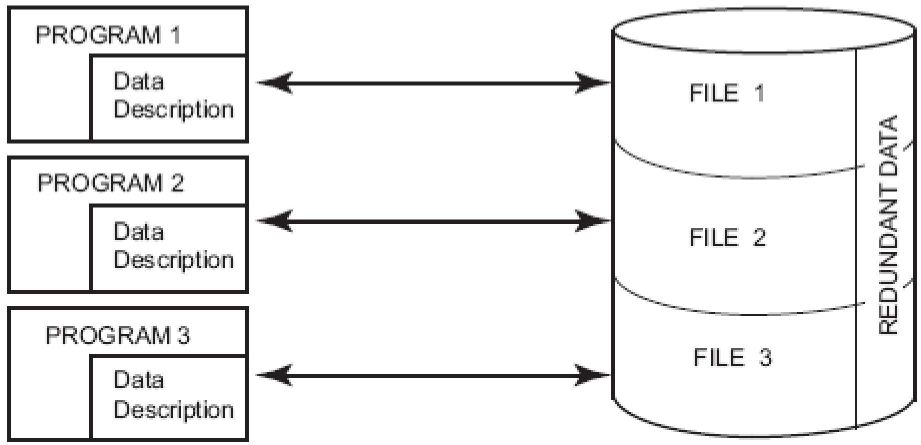
- Applications are inherently distributed.
- A distributed system is more reliable.
- A distributed system performs better.
- A distributed system scales better.

Distributed Database Systems

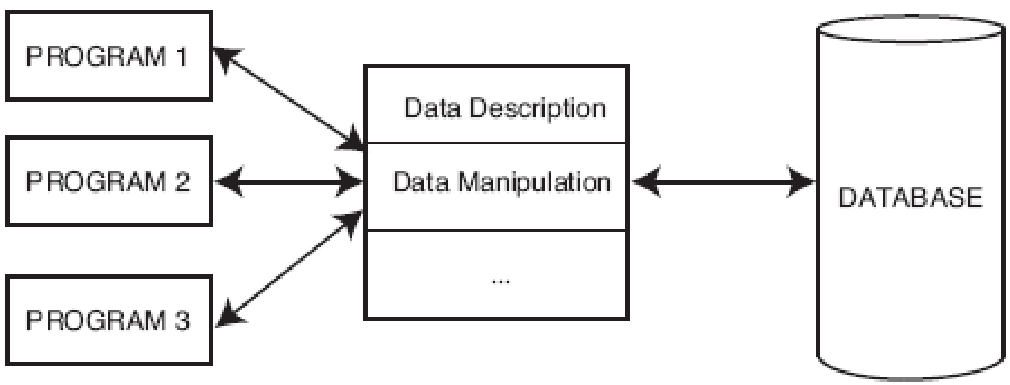
- Union of two technologies:
  - Database Systems + Computer Networks
- Database systems provide
  - data independence (physical & logical)
  - centralized and controlled data access
  - integration
- Computer networks provide distribution.
- integration ≠ centralization
- integration + distribution

DBMS Provides Data Independence

File Systems



Database Management Systems



Distributed Database Systems

- Union of two technologies:
  - Database Systems + Computer Networks
- Database systems provide
  - data independence (physical & logical)
  - centralized and controlled data access
  - integration
- Computer networks provide distribution.
- integration ≠ centralization
- integration + distribution

Distributed Systems

- Tanenbaum et al:

  "*a collection of <u>independent</u> computers that appears to its users as a <u>single coherent</u> system*"

- Coulouris et al:

  "*a system in which hardware and software components located at <u>networked computers</u> communicate and coordinate their actions only by <u>passing messages</u>*"

Distributed Systems

- Ozsu et al:

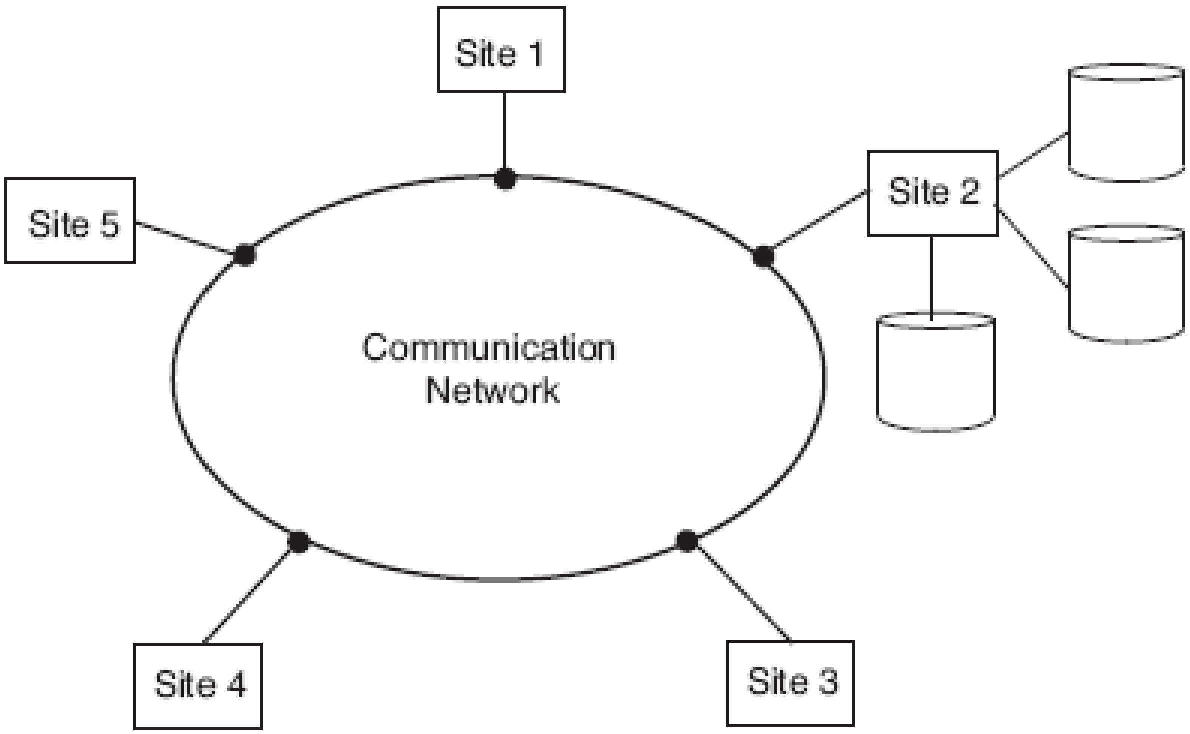
  "a number of <u>autonomous</u> processing elements (<u>not necessarily homogeneous</u>) that are <u>interconnected</u> by a computer network and that <u>cooperate</u> in performing their assigned tasks"

What is being distributed?

- Processing logic
- Function
- Data
- Control

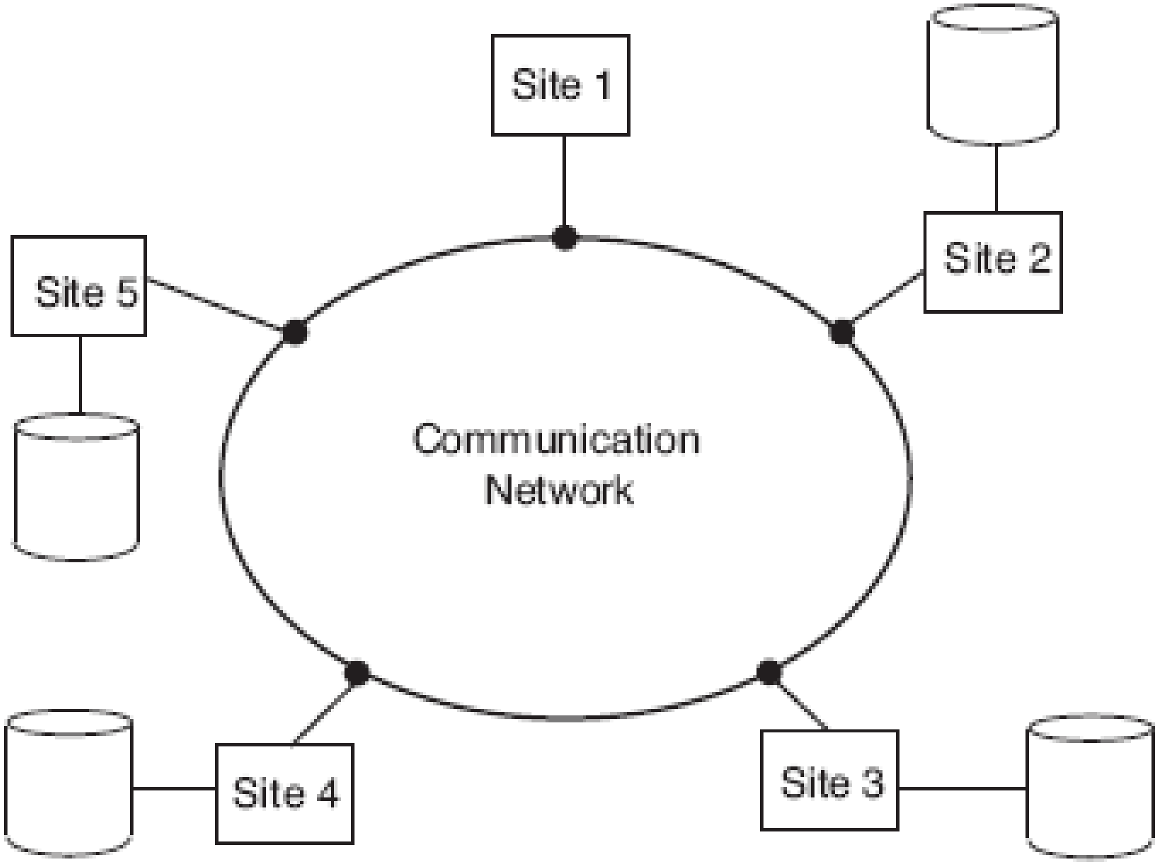
- For distributed DBMSs, all are required.

Centralized DBMS on a Network



What is being distributed here?

Distributed DBMS



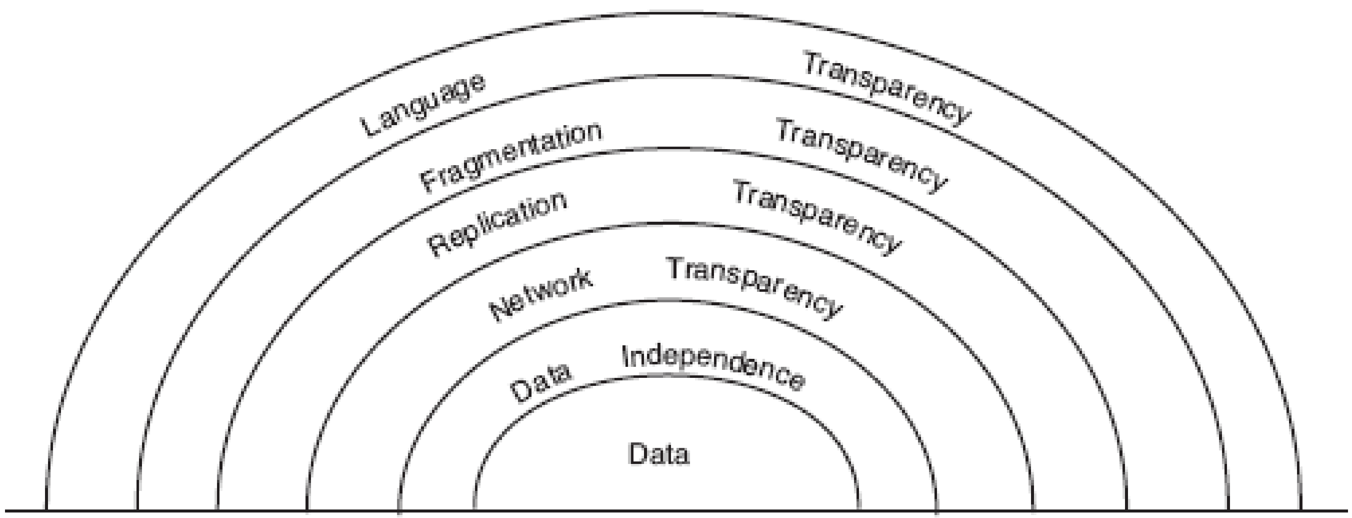
And here?

Distributed DBMS Promises

1. Transparent management of distributed and replicated data
2. Reliability/availability through distributed transactions
3. Improved performance
4. Easier and more economical system expansion

Promise #1: Transparency

- Hiding implementation details from users
- Providing **data independence** in the distributed environment
- Different transparency types, related:



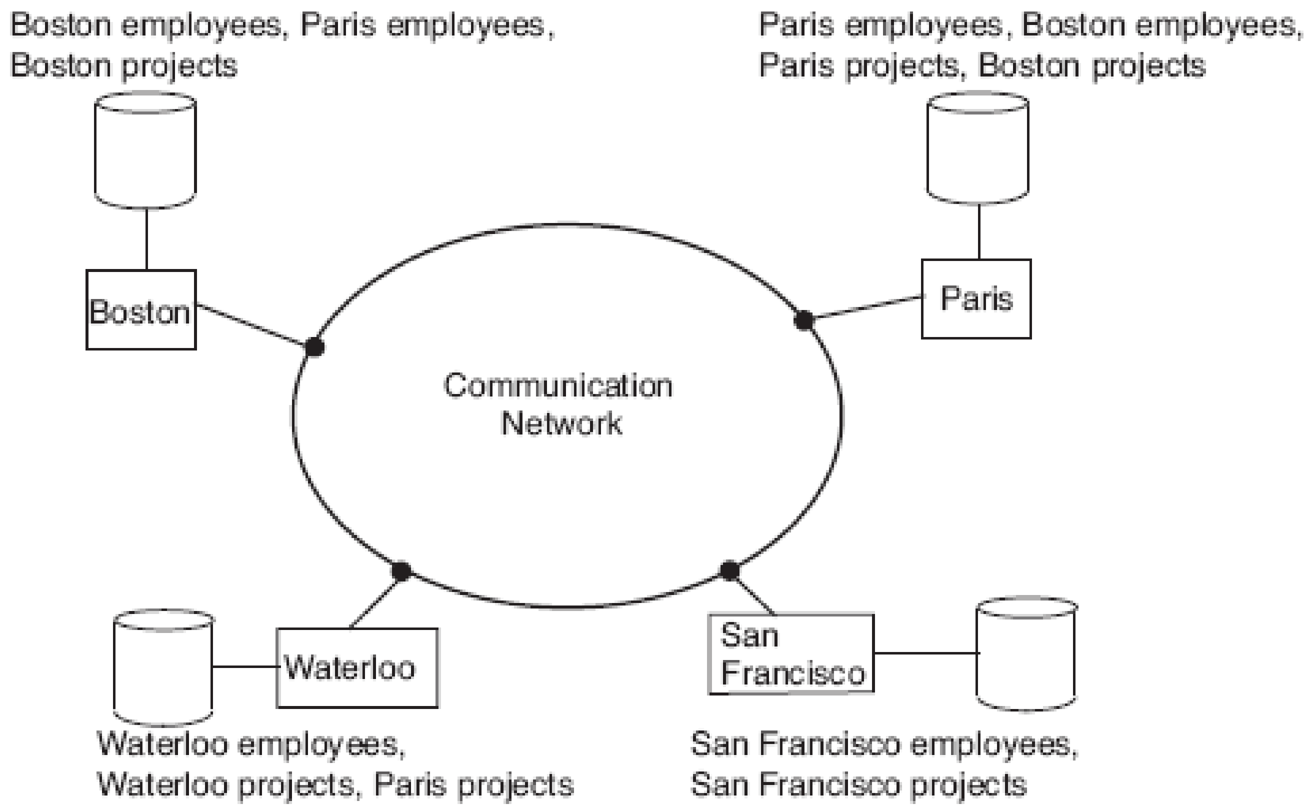
- Full transparency is neither always possible nor desirable!

Transparency Example

- Employee (eno, ename, title)
- Project (pno, pname, budget)
- Salary (title, amount)
- Assignment (eno, pno, responsibility, duration)

```
SELECT   ename, amount
FROM     Employee, Assignment, Salary
WHERE    Assigment.duration > 12
AND      Employee.eno = Assignment.eno
AND      Salary.title = Employee.title
```

Transparency Example



What types of transparencies are provided here?

Promise #2: Reliability & Availability

- Distribution of replicated components
- When sites or links between sites fail
  - No single point of failure
- Distributed transaction protocols keep database consistent via
  - Concurrency transparency
  - Failure atomicity

Promise #3: Improved Performance

- Place data fragments closer to their users
  - less contention for CPU and I/O at a given site
  - reduced remote access delay
- Exploit parallelism in execution
  - inter-query parallelism
  - intra-query parallelism

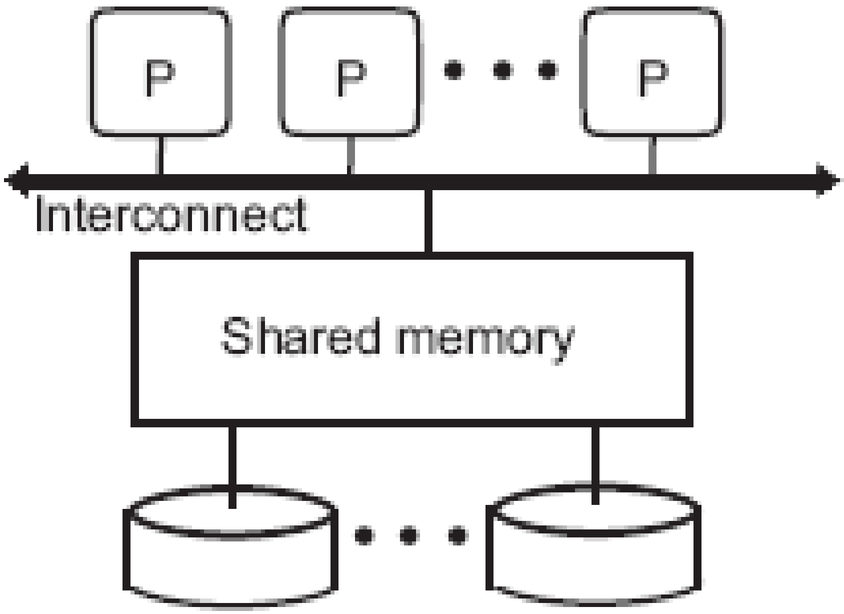
Promise #4: Easy Expansion

- It is easier to scale a distributed collection of smaller systems than one big centralized system.

How do we distribute?

- Basic distributed architectures:
  - Shared-Memory
  - Shared-Disk
  - Shared-Nothing

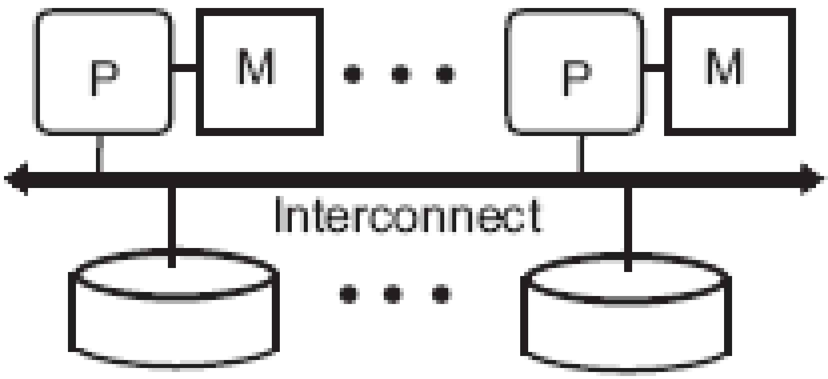
Shared-Memory



- Fast interconnect
- Single OS

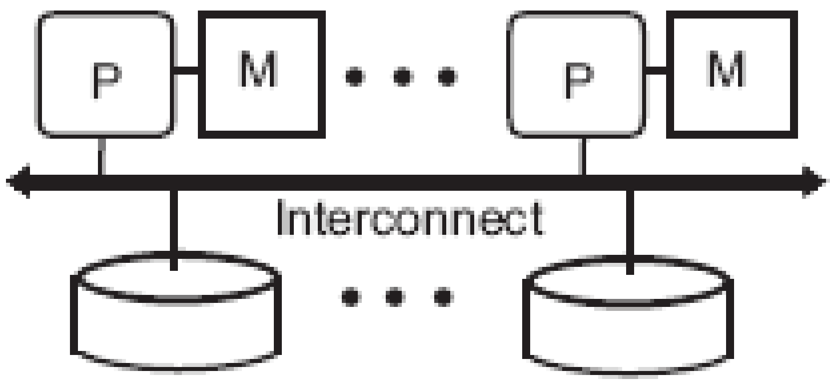
- Advantages:
  – Simplicity
  – Easy load balancing
- Problems:
  – High cost (the interconnect)
  – Limited extensibility (~ 10)
  – Low availability

Shared-Disk



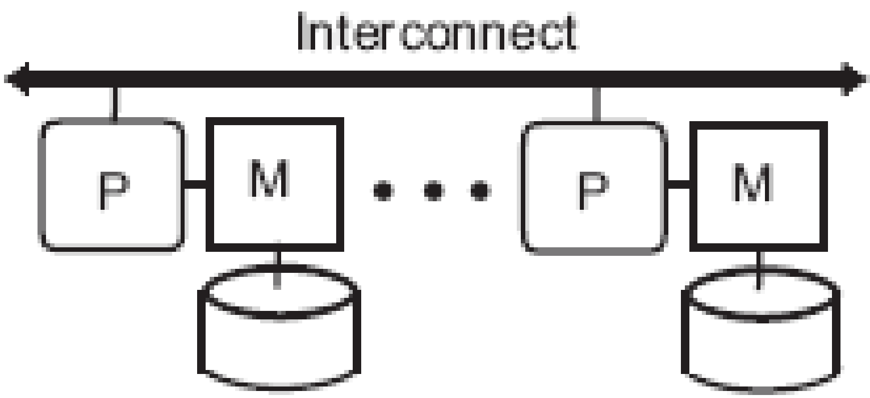
- Separate OS per P-M
- Advantages:
  - No distributed database design - easy migration/evolution
  - Load balancing
  - Availability
- Problems:
  - Limited extensibility (~ 20) - disk/interconnect bottleneck

Shared-Cache



- Oracle RAC
- Interconnect is used to communicate between nodes and disk:
  if data are missing in the local buffer, they are first queried in buffers on other nodes and then on the disk
- The same pros/cons, just faster

Shared-Nothing



- Separate OS per P-M-D
- E.g. DB2 Parallel Edition, Teradata

- Advantages:
  - Extensibility and scalability
  - Lower cost
  - High availability
- Problems:
  - Distributed database design for particular queries/workload

Retrospective summary

- Shared-cache (disk) won in enterprise because:
  - enterprises usually do not requires extreme scalability
  - it was easy to migrate from non-distributed database
- Shared-Nothing is now popular because of the Web applications require extreme scalability

Basic Shared-Nothing Techniques

- Data Partitioning
- Data Replication
- Query Decomposition and Function Shipping

Shared-Nothing Techniques: Partitioning

- Each relation is divided into n partitions that are mapped onto different disks.
- Provides storing large amounts of data and improved performance
- By key - values of a column(s):
  - Range
    - e.g. using B-tree index
    - Supports range queries but index required
  - Hashing
    - Hash function
    - Only exact-match queries but no index
- Provides storing large amounts of data and improved performance

Shared-Nothing Techniques: Replication

- Storing copies of data on different nodes
- Provides high availability and reliability
- Requires distributed transactions to keep replicas consistent:
  - Two phase commit - data always consistent but the system is fragile
  - Eventually consistency - eventually becomes consistent but always writable

Shared-Nothing Techniques: Query Decomposition and Shipping

- Query operations are performed where the data resides.
  - Query is decomposed into subtasks according to the data placement (partitioning and replication).
  - Subtasks are executed at the corresponding nodes.
- Data placement is always good only for some queries =>
  - hard to design database
  - need to redesign when queries change

Classes of shared-nothing databases

- Two broad classes of shared-nothing systems we will talk about:
  - SQL DBMS - DB2 Parallel Edition (Enterprise apps)
  - Key-value store - Cassandra (Web apps)

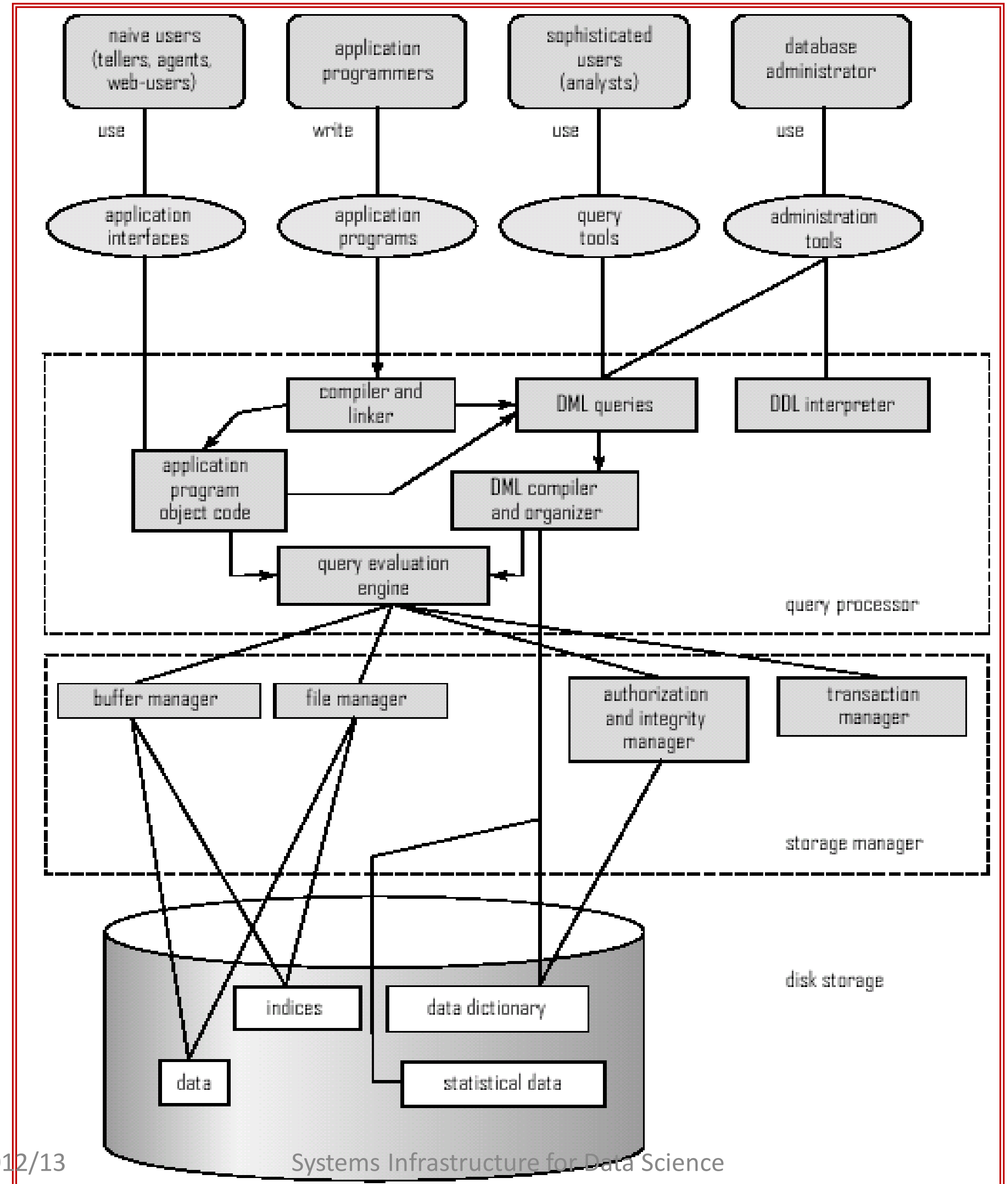
Distributed DBMS Major Design Issues

- Distributed DB design (Data storage)
  - partition vs. replicate
  - full vs. partial replicas
  - optimal fragmentation and distribution is NP-hard
- Distributed metadata management
  - where to place directory data
- Distributed query processing
  - cost-efficient query execution over the network
  - query optimization is NP-hard

Distributed DBMS Major Design Issues

- Distributed transaction management
  - Synchronizing concurrent access
  - Consistency of multiple copies of data
  - Detecting and recovering from failures
  - Deadlock management
  - Providing ACID properties in general

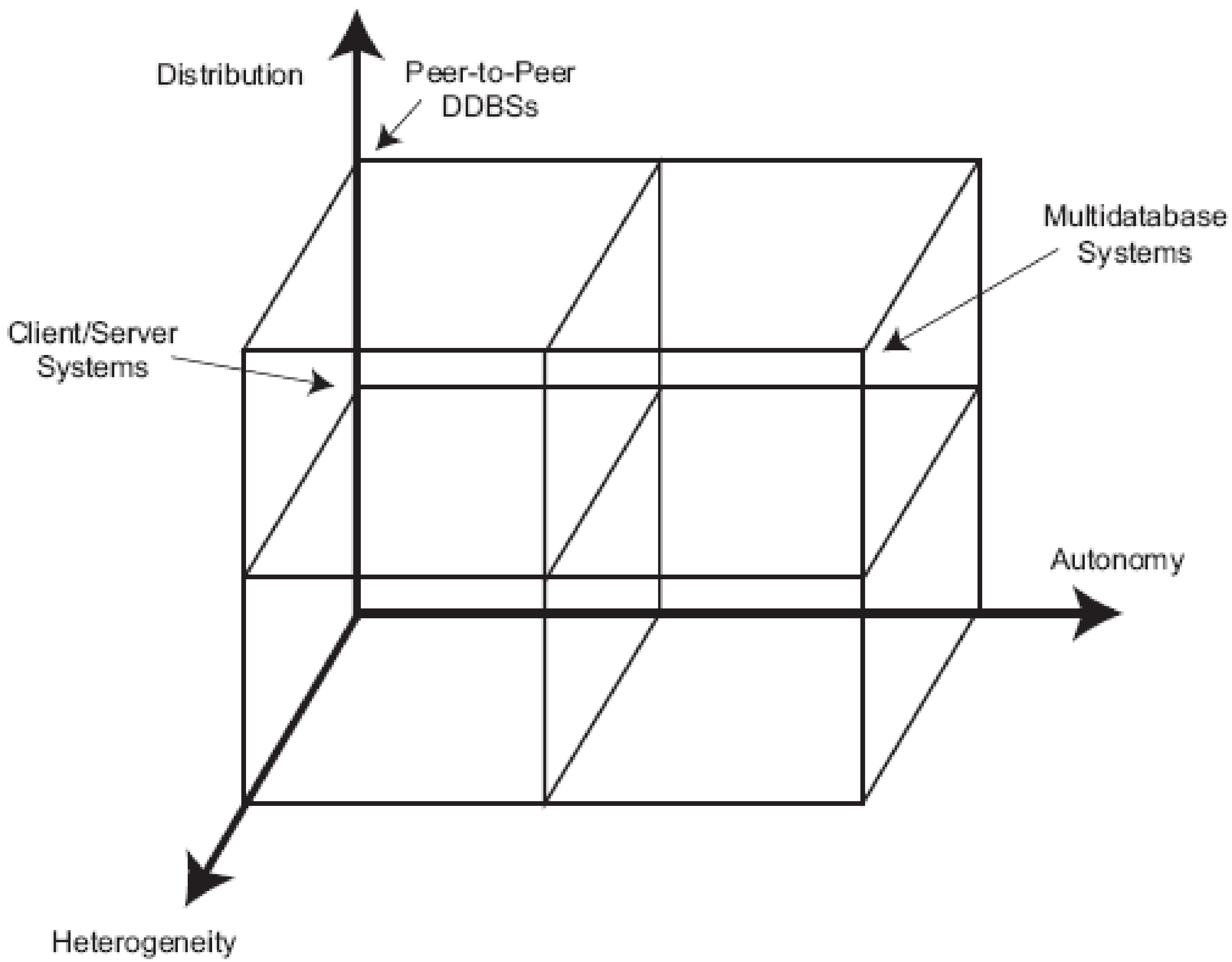
=> Distributed Systems Lecture (Schindelhauer/Lausen)



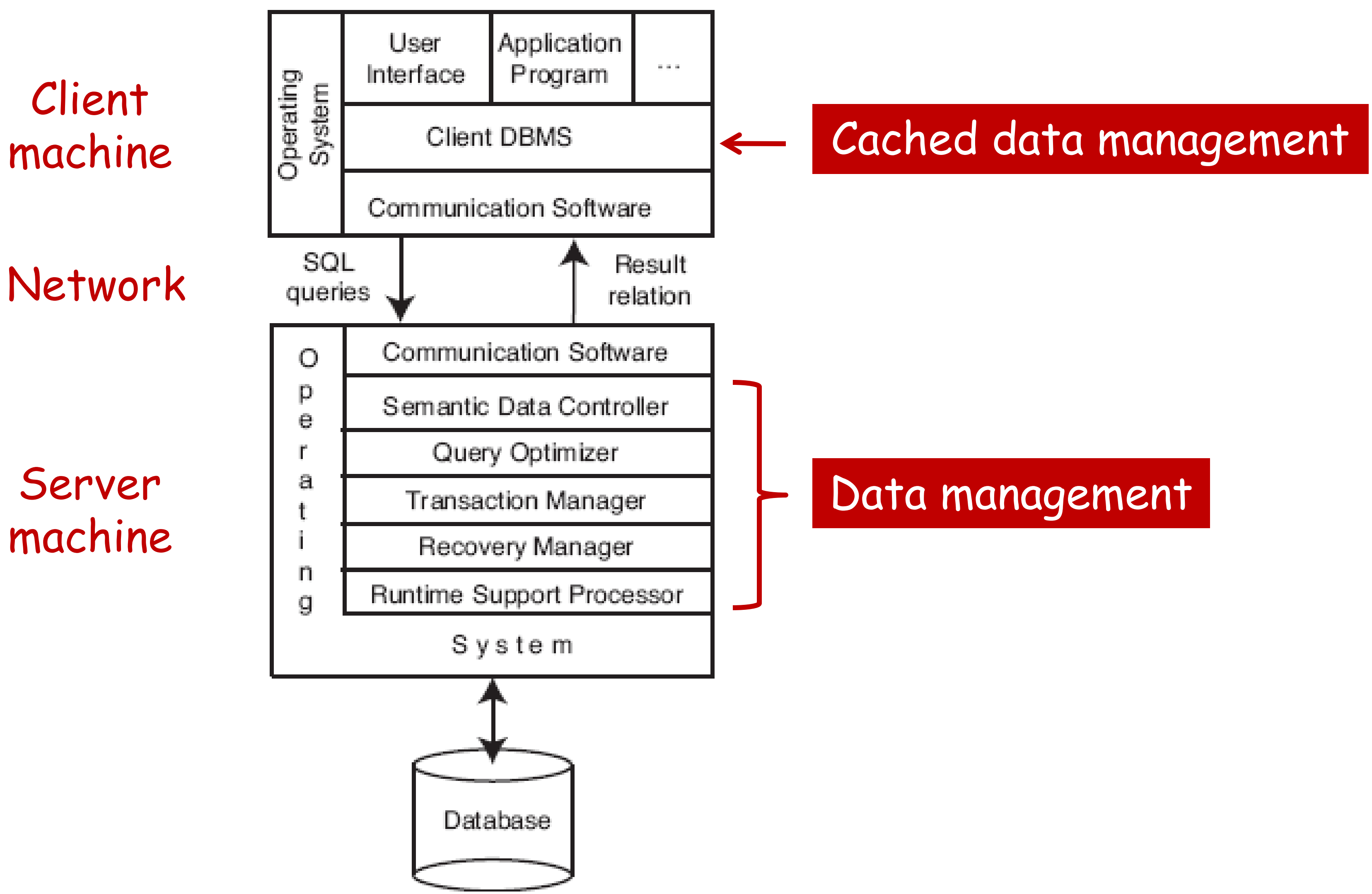
Typical Centralized DBMS Architecture

[Silberschatz et al]

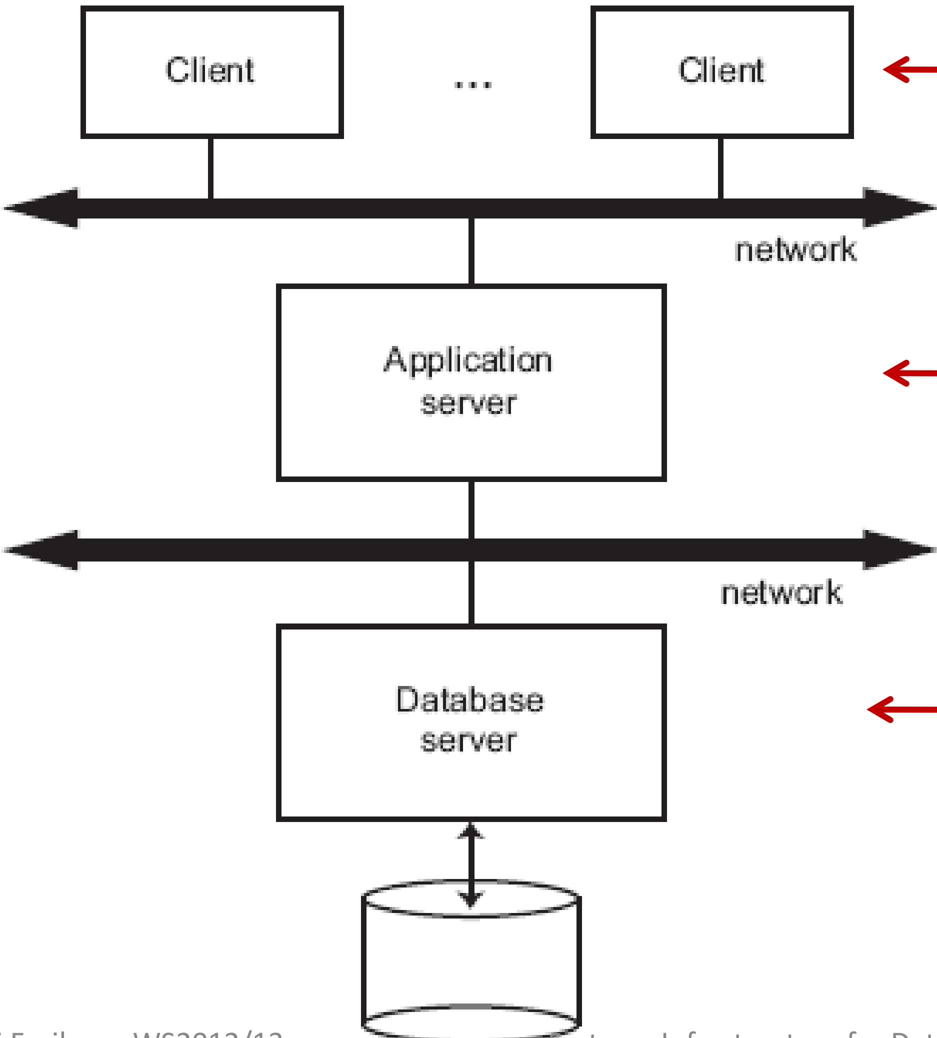
Important Architectural Dimensions for Distributed DBMSs



Client/Server DBMS Architecture



Three-tier Client/Server Architecture



User interface

Application programs

Data management

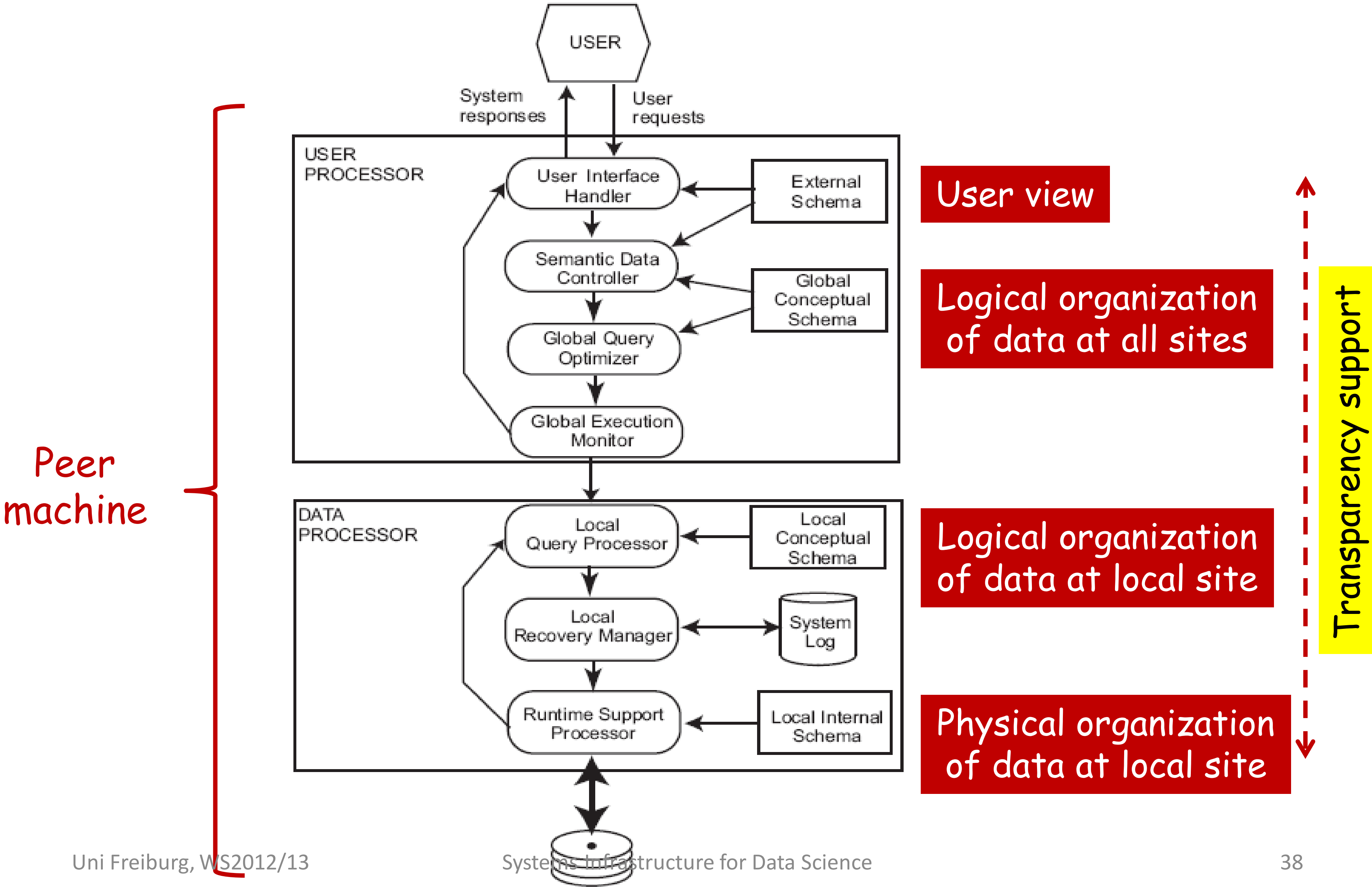
Extensions to Client/Server Architectures

- Multiple clients
- Multiple application servers
- Multiple database servers

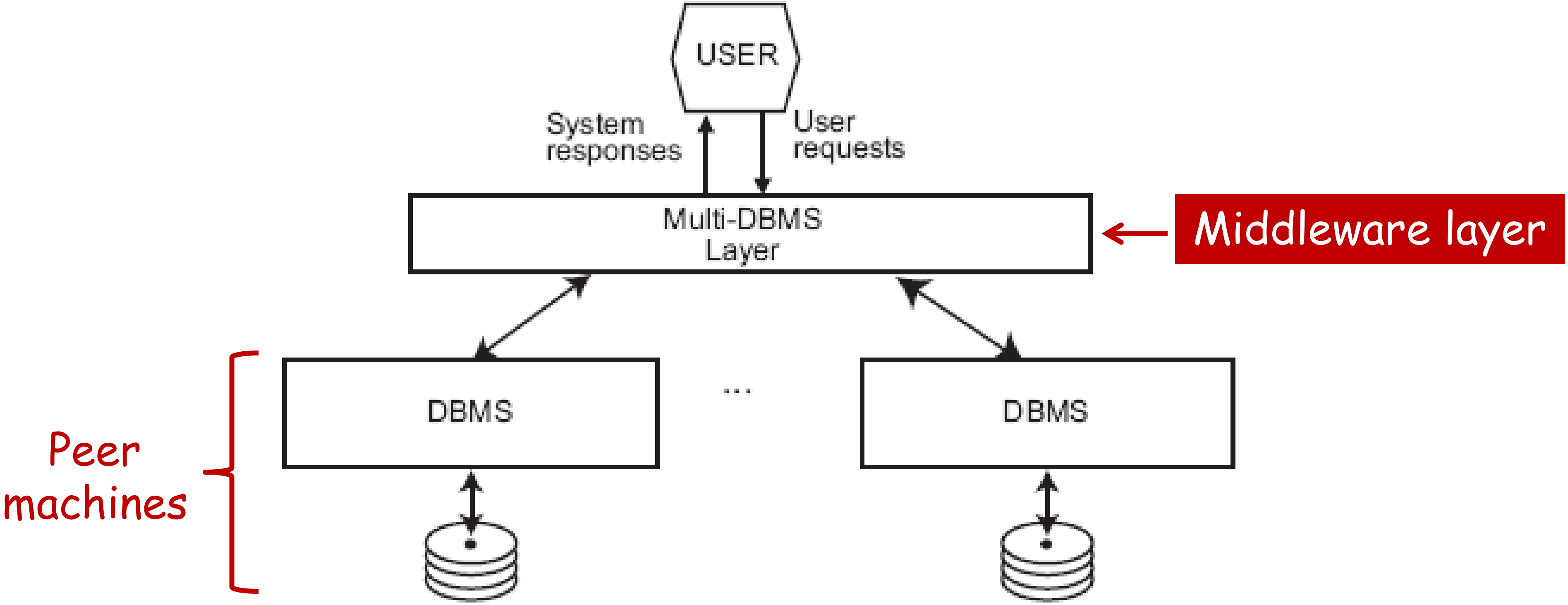
Peer-to-Peer DBMS Systems

- Classical (same functionality at each site)
- Modern (as in P2P data sharing systems)
  - Large scale
  - Massive distribution
  - High heterogeneity
  - High autonomy

Classical Peer-to-Peer DBMS Architecture



Multi-database System Architecture



- Full autonomy
- Potential heterogeneity

What is a Distributed DBMS?

- Distributed database:
  - *"a collection of multiple, logically interrelated databases distributed over a computer network"*
- Distributed DBMS:
  - *"the software system that permits the management of the distributed database and makes the distribution transparent to the users"*
- This definition is relaxed for modern networked information systems (e.g., web).