# PRINCIPLES AND TECHNOLOGIES OF DISTRIBUTED **DATABASE** SYSTEM – INTRODUCTION

李旭东 **Li-Xudong**

LEEXUDONG@NANKAI.EDU.CN

NANKAI UNIVERSITY

# OBJECTIVES

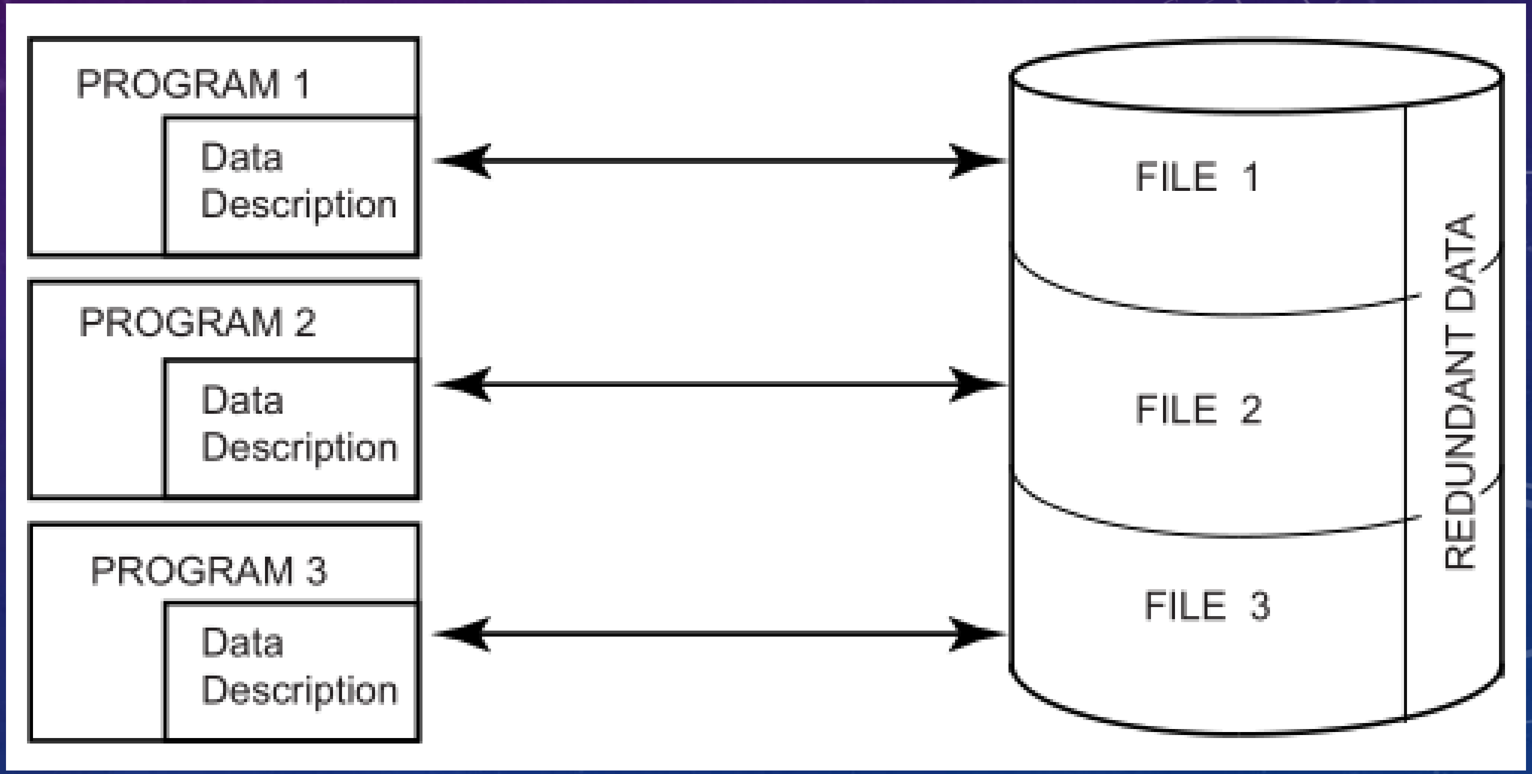
- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# OBJECTIVES

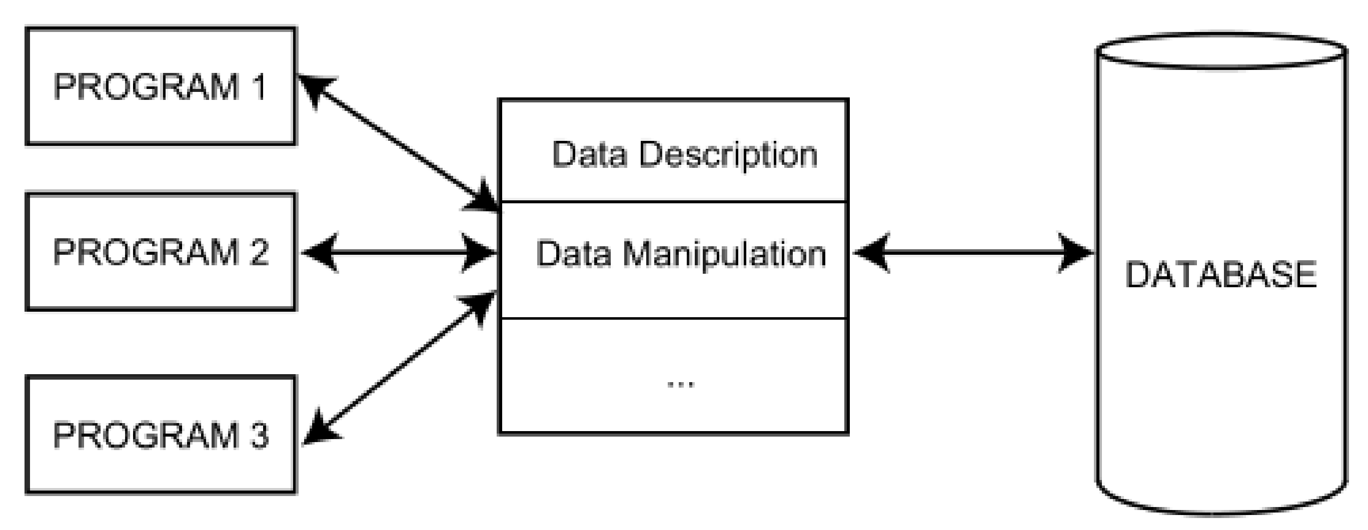
- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# THE AGE OF DATABASE

- Data
- File system



# THE AGE OF DATABASE




- Data
- File system
- Network
- Data Independence(数据独立性)
- Database system

# DATABASE SYSTEM

- Database-management system
  - DBMS
  - A collection of interrelated data and a set of programs to access those data
- Goal of DBMS
  - Provide a way to store and retrieve database information that is both convenient and efficient

# DATABASE-SYSTEM APPLICATIONS

- Enterprise Information
  - Sales, accounting, human resources, manufacturing, online retailers
- Banking and Finance
  - Banking, credit card transactions, finance stock
- Universities
- Airlines
- Telecommunication
- …

# PURPOSE OF DATABASE SYSTEMS

- Enable durability (持久性) storage
- Large amounts of data
- Isolation (独立性) among users
- Atomicity(原子性)
- Reduce redundancy(冗余)
- Consistency(一致性)
- Integrity (完整性)
- Efficient store and access
- Concurrent(并发) access
- Security(安全)

# DATA : CASE

```
type instructor = record
    ID : string;
    name : string;
    dept_name : string;
    salary : integer;
   birth : date;
   gender : boolean;
end;
```

```
type department = record
  …
end;

type student = record
  …
end;
```

# DATA MODEL

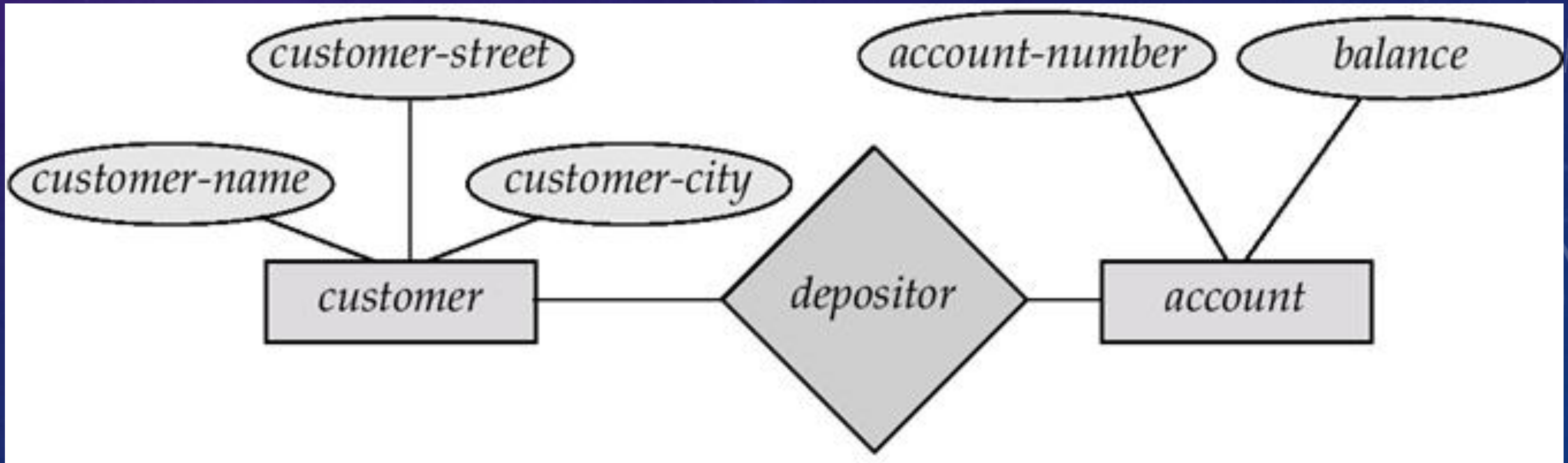
- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints

# DATA （DATABASE） MODEL

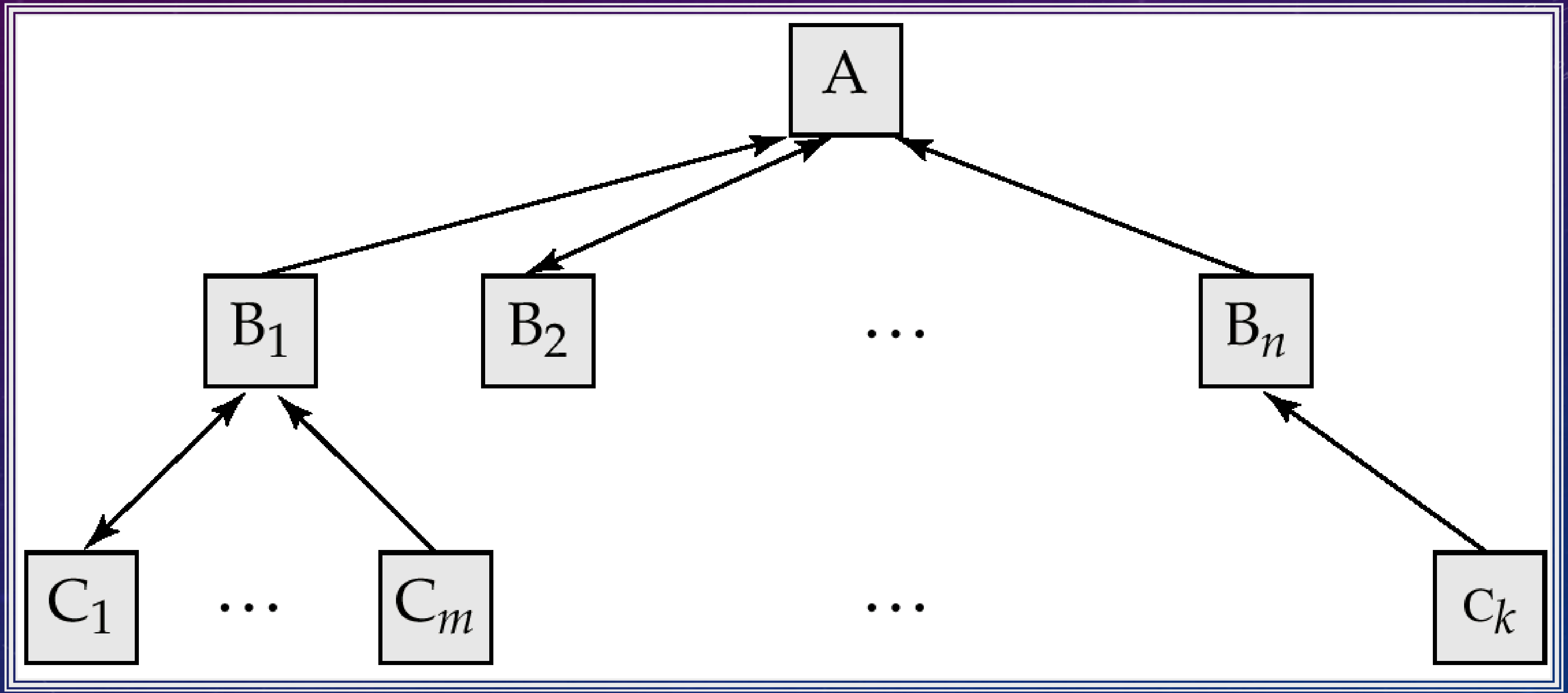
- Entity-relationship model(实体关系E-R)
- Hierarchical data model(层次)
- Network data model(网络)
- Relational data model(关系)
- Object-based data model(对象)
- Semi-structured data model (半结构)
- Graph data model(图), …

# ENTITY-RELATIONSHIP MODEL

- Entity-relationship model
  - 实体关系模型E-R
  - Entity
  - Relationship
  - Attribute属性



# HIERARCHICAL DATA MODEL



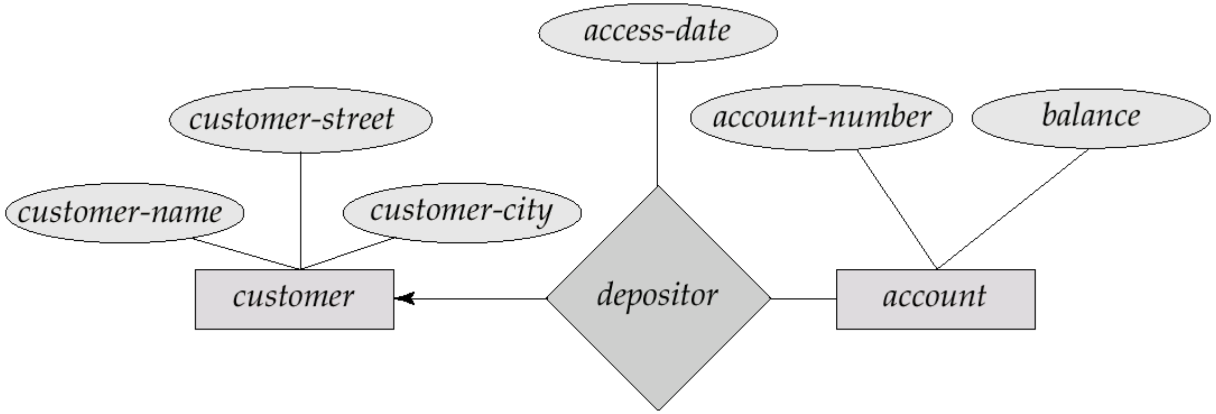
# HIERARCHICAL DATA MODEL

- A hierarchical database consists of a collection of *records* which are connected to one another through *links.*
- a record is a collection of fields, each of which contains only one data value.
- A link is an association between precisely two records.
- The hierarchical model differs from the network model in that the records are organized as collections of trees rather than as arbitrary graphs.

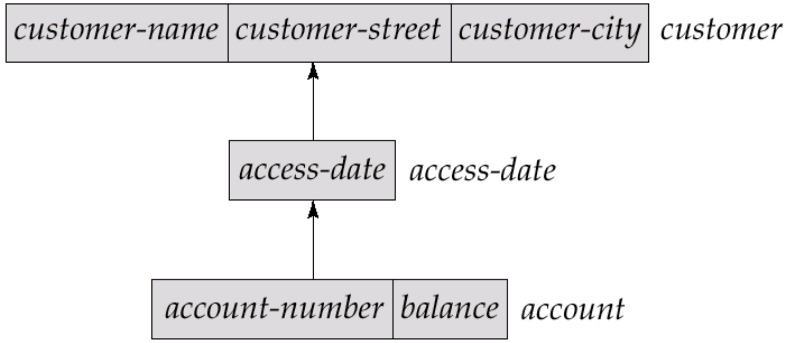
# HIERARCHICAL DATA MODEL

- The schema for a hierarchical database consists of
  - *boxes,* which correspond to record types
  - *lines,* which correspond to links
- Record types are organized in the form of a *rooted tree.*
  - **No cycles** in the underlying graph.
  - Relationships formed in the graph must be such that only one-to-many or one-to-one relationships exist between a parent and a child.

# HIERARCHICAL DATA MODEL: CASE




(a) E-R diagram




(b) Tree-structure diagram

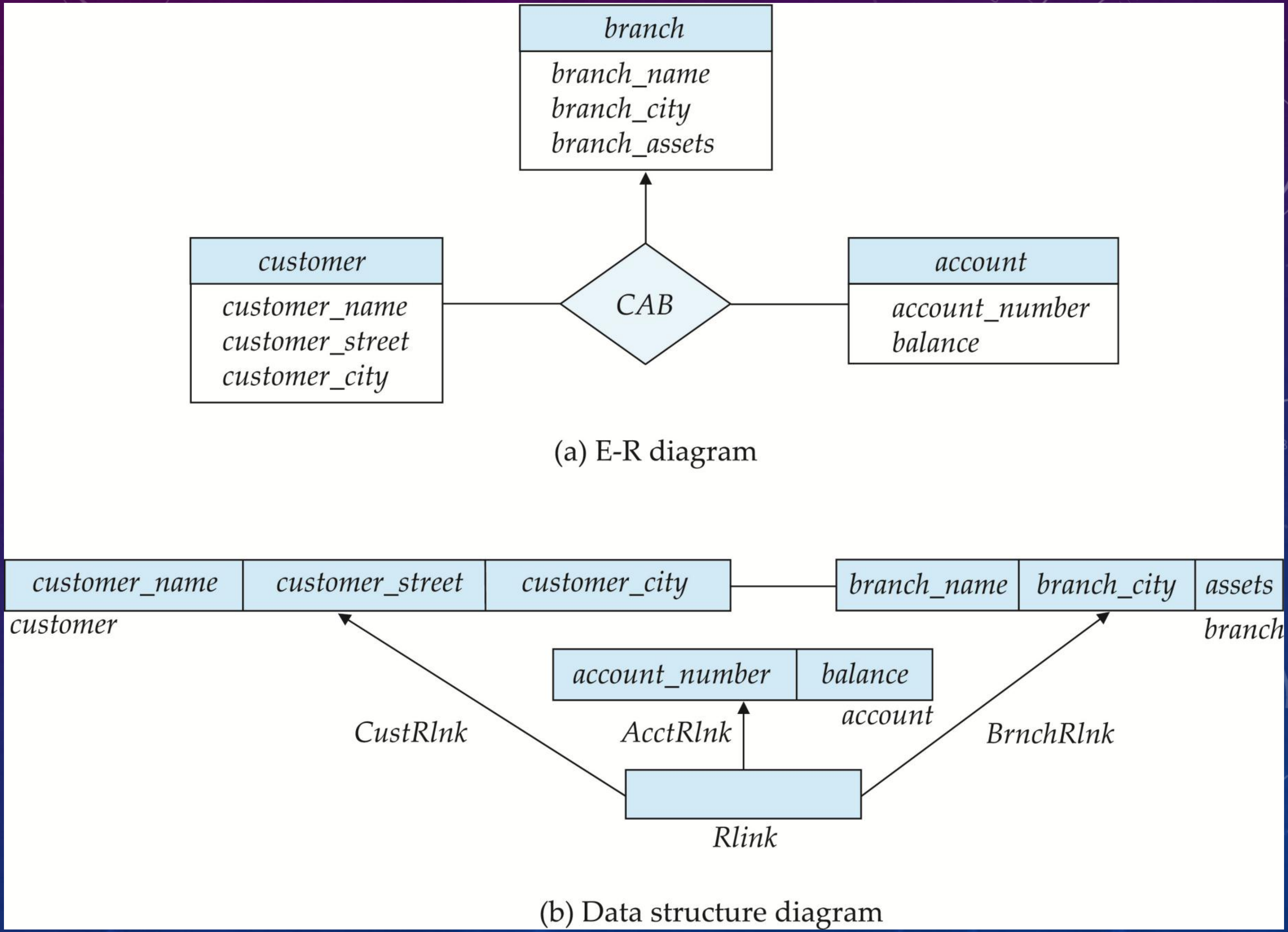
# NETWORK DATA MODEL

- Data are represented by collections of *records*.
  - similar to an entity in the E-R model
  - Records and their fields are represented as *record type*
- Relationships among data are represented by *links*
  - similar to a restricted (binary) form of an E-R relationship
  - restrictions on links depend on whether the relationship is many-many, many-to-one, or one-to-one.

# NETWORK DATA MODEL

- Schema representing the design of a network database.
- A data-structure diagram consists of two basic components:
  - **Boxes**, which correspond to record types.
  - **Lines**, which correspond to links.
- Specifies the overall logical structure of the database.

# NETWORK DATA MODEL：CASE



# RELATIONAL DATA MODEL

- Relational data model
  - A collection of tables
  - Table
    - A collection of records
    - Multiple columns
  - Relation

# RELATIONAL DATA MODEL: CASE

Columns

Rows

| customer_id | customer_name | customer_street | customer_city | account_number |
|---|---|---|---|---|
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto | A-101 |
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto | A-201 |
| 677-89-9011 | Hayes | 3 Main St. | Harrison | A-102 |
| 182-73-6091 | Turner | 123 Putnam St. | Stamford | A-305 |
| 321-12-3123 | Jones | 100 Main St. | Harrison | A-217 |
| 336-66-9999 | Lindsay | 175 Park Ave. | Pittsfield | A-222 |
| 019-28-3746 | Smith | 72 North St. | Rye | A-201 |

# RELATIONAL DATA MODEL: CASE

| *customer-id* | *customer-name* | *customer-street* | *customer-city* |
|---|---|---|---|
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto |
| 019-28-3746 | Smith | 4 North St. | Rye |
| 677-89-9011 | Hayes | 3 Main St. | Harrison |
| 182-73-6091 | Turner | 123 Putnam Ave. | Stamford |
| 321-12-3123 | Jones | 100 Main St. | Harrison |
| 336-66-9999 | Lindsay | 175 Park Ave. | Pittsfield |
| 019-28-3746 | Smith | 72 North St. | Rye |

(a) The *customer* table

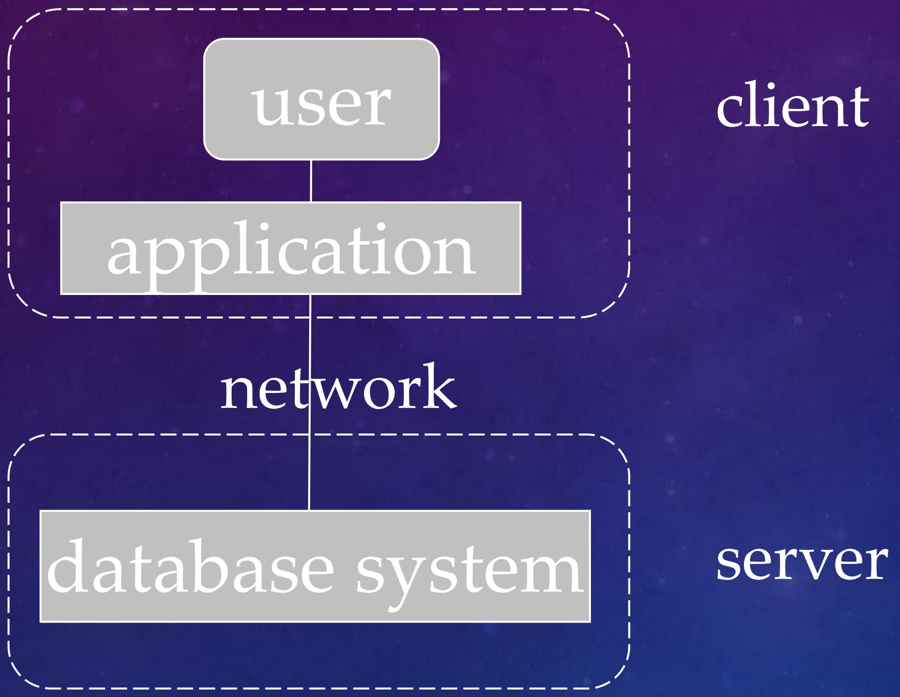
| *account-number* | *balance* |
|---|---|
| A-101 | 500 |
| A-215 | 700 |
| A-102 | 400 |
| A-305 | 350 |
| A-201 | 900 |
| A-217 | 750 |
| A-222 | 700 |

(b) The *account* table

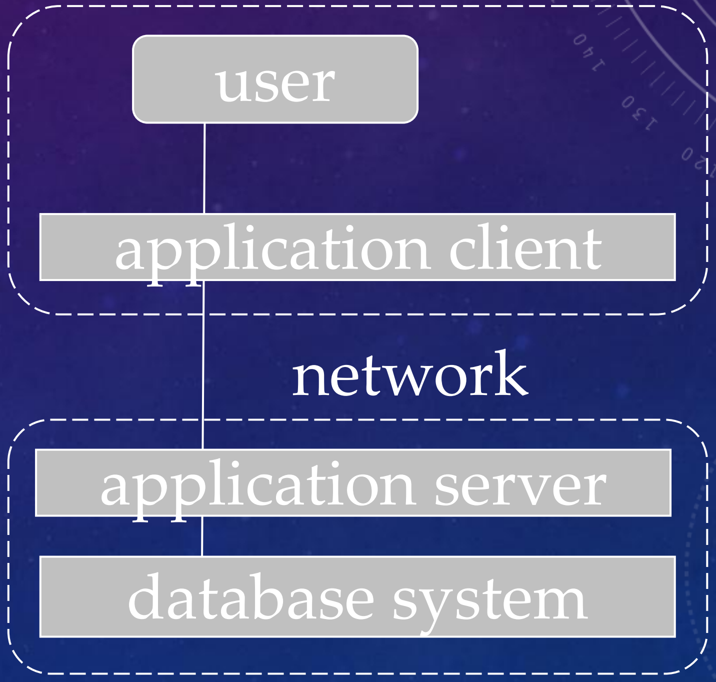
| *customer-id* | *account-number* |
|---|---|
| 192-83-7465 | A-101 |
| 192-83-7465 | A-201 |
| 019-28-3746 | A-215 |
| 677-89-9011 | A-102 |
| 182-73-6091 | A-305 |
| 321-12-3123 | A-217 |
| 336-66-9999 | A-222 |
| 019-28-3746 | A-201 |

(c) The *depositor* table

# ARCHITECTURE OF APPLICATION BASED ON CENTRALIZED DATABASE

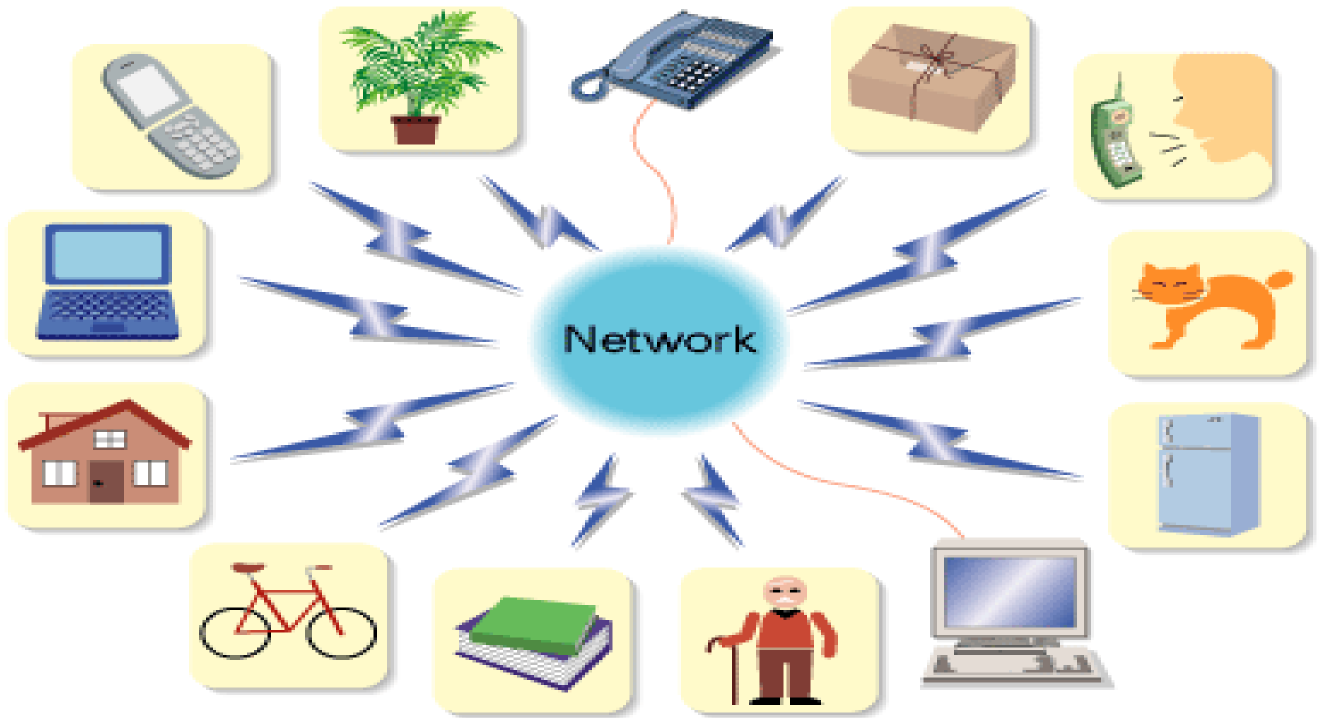



(a) Two-tier architecture



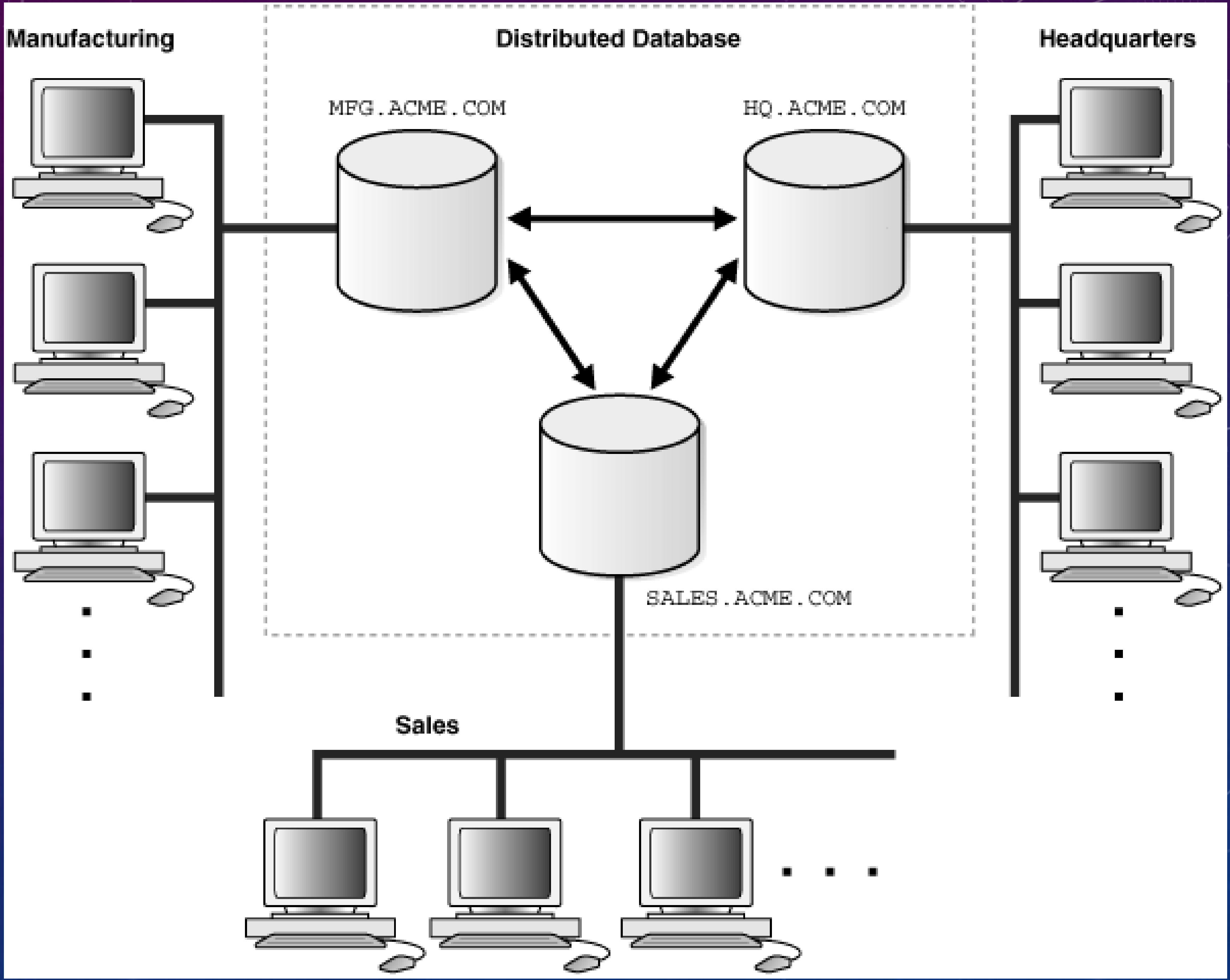

(b) Three-tier architecture

# DIGITAL DATA EXPLOSION …

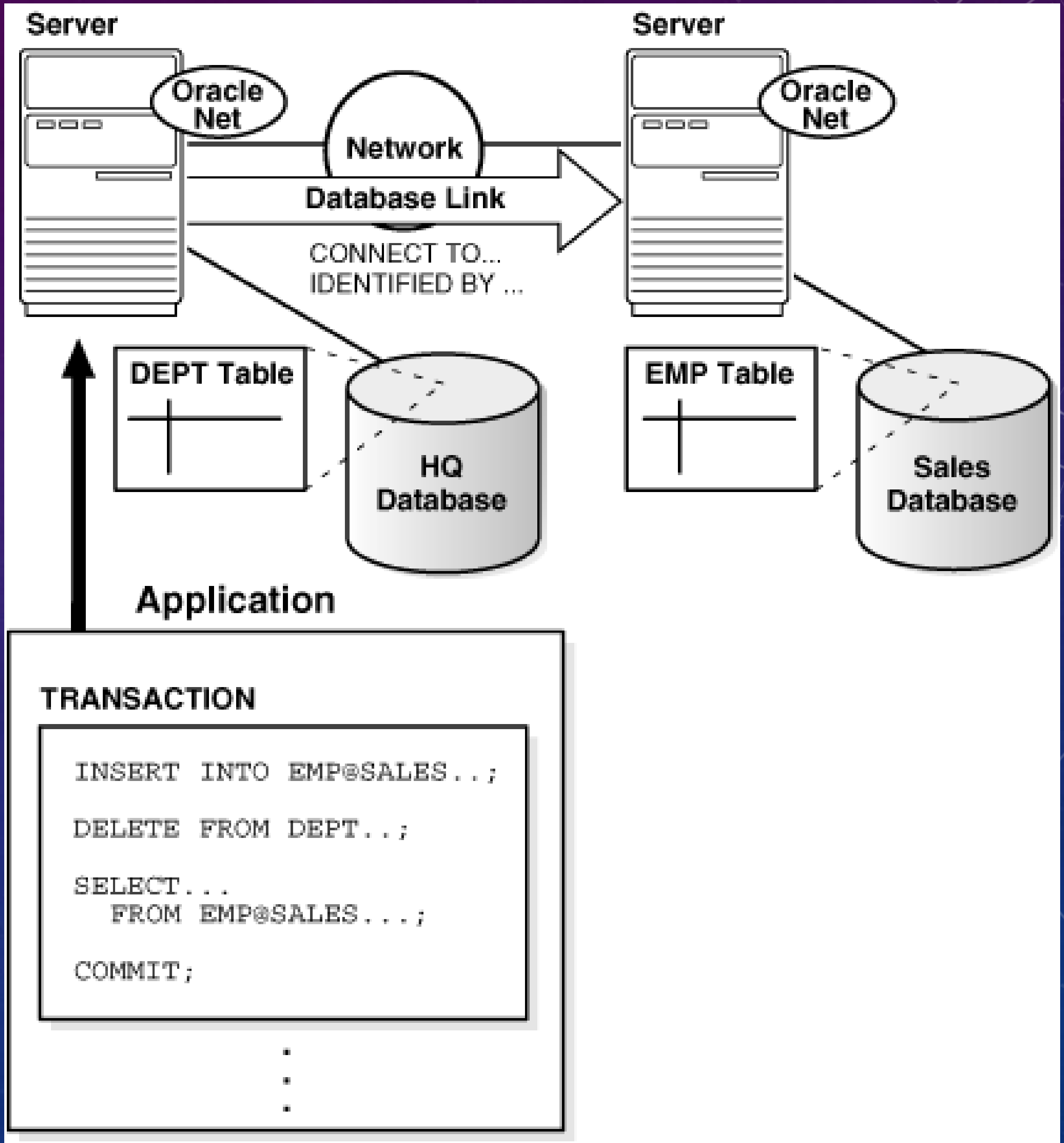



Ubiquitous computing will enable diverse wireless applications, including monitoring of pets and houseplants, operation of appliances, keeping track of books and bicycles, and much more.

# DISTRIBUTED DATABASE



# DISTRIBUTED DATABASE (ORACLE)



The most important objective of the database technology is **integration**(集成), not **centralization**(集中)

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# DISTRIBUTED DATA PROCESSING

- Distributed processing, Distributed computing
- Distributed computing system
  - It is a number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks.
  - The "processing element" referred to in this definition is a computing device that can execute a program on its own.

# DISTRIBUTED DATA PROCESSING – WHAT IS BEING DISTRIBUTED?

- (1) processing logic处理逻辑 is distributed
  - the definition of a distributed computing system given above implicitly assumes that the processing logic or processing elements are distributed

# DISTRIBUTED DATA PROCESSING – WHAT IS BEING DISTRIBUTED?

- (2) function功能 is distributed
  - Various functions of a computer system could be delegated to various pieces of hardware or software

# DISTRIBUTED DATA PROCESSING – WHAT IS BEING DISTRIBUTED?

- (3) data数据 is distributed
  - Data used by a number of applications may be distributed to a number of processing sites

# DISTRIBUTED DATA PROCESSING – WHAT IS BEING DISTRIBUTED?

- (4) control控制 is distributed
  - The control of the execution of various tasks might be distributed instead of being performed by one computer system.

# DISTRIBUTED DATA PROCESSING – WHY DO WE DISTRIBUTE AT ALL?

- (1) The classical answers to this question indicate that **distributed processing** better corresponds to the organizational structure of today's widely distributed enterprises, and that such a system is more reliable and more responsive.
- More importantly, many of the current applications of computer technology are inherently distributed.

# DISTRIBUTED DATA PROCESSING – WHY DO WE DISTRIBUTE AT ALL?

- (2) it can be stated that the fundamental reason behind distributed processing is to be better able to cope with the large-scale data management problems that we face today, by using a variation of the well-known **divide-and-conquer rule**分而治之.

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

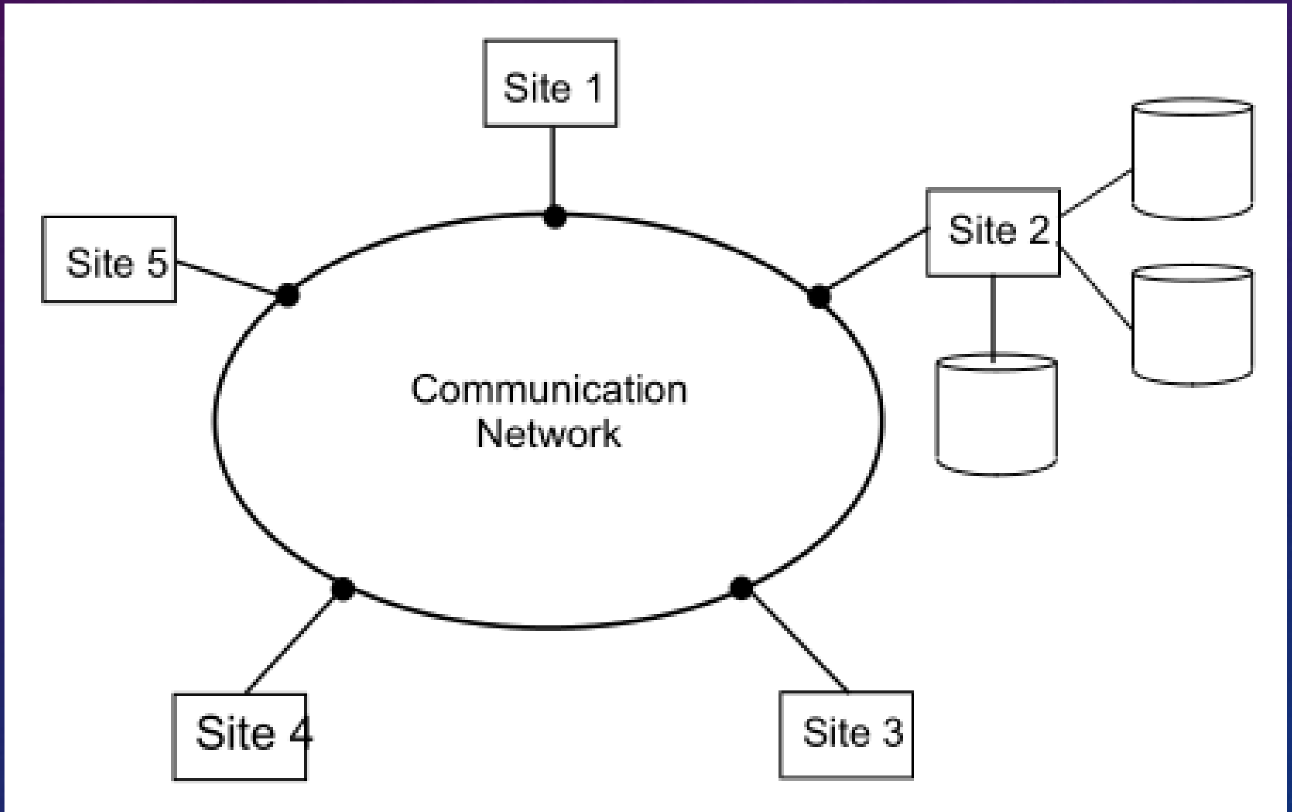
# DISTRIBUTED DATABASE SYSTEM (DDBS)

- Distributed database system (DDBS)
  - a collection of multiple, logically interrelated databases distributed over a computer network
    - (1) the distributed database
    - (2) the distributed DBMS

# DISTRIBUTED DATABASE SYSTEM (DDBS)

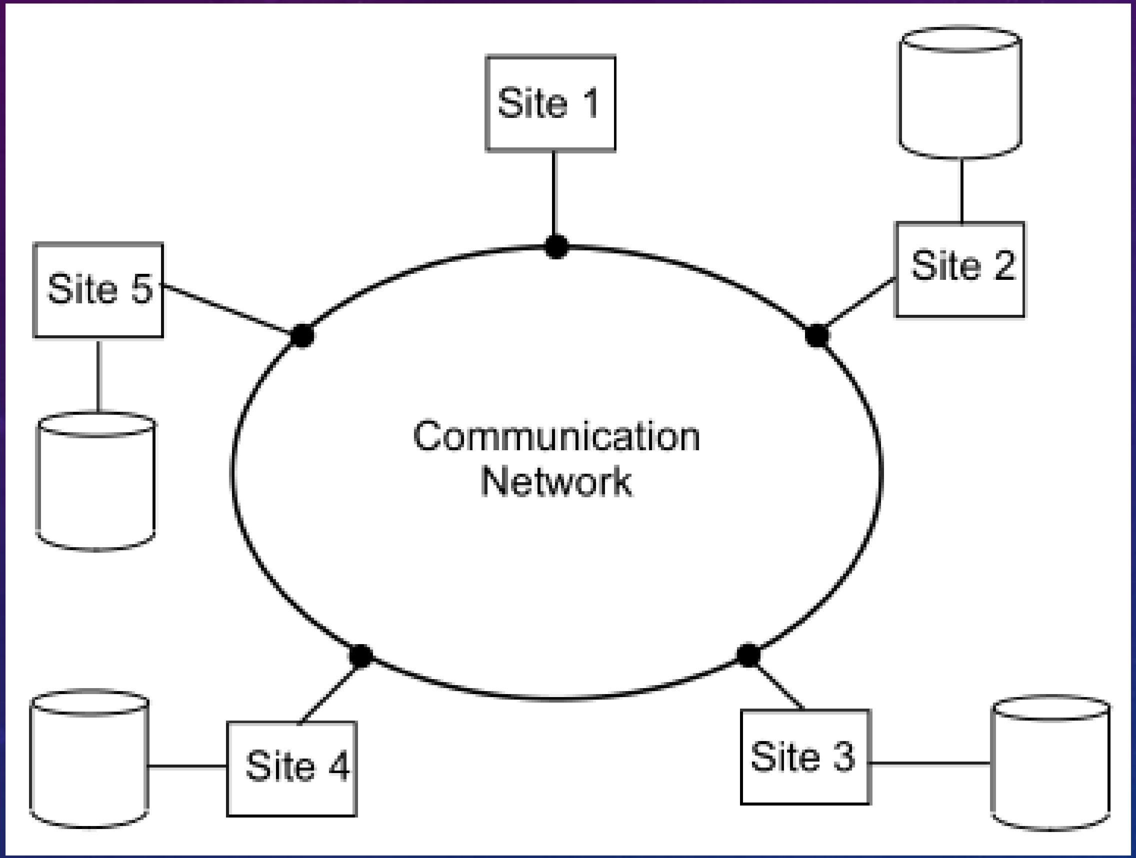
- the physical distribution of data is important
  - Note that physical distribution does not necessarily imply that the computer systems be geographically far apart; they could actually be in the same room.
  - It simply implies that the communication between them is done over a network instead of through shared memory or shared disk (as would be the case with multiprocessor systems), with the network as the only shared resource.

# DISTRIBUTED DATABASE SYSTEM (DDBS)




Central Database on a Network

# DISTRIBUTED DATABASE SYSTEM (DDBS)




DDBS Environment

- logically interrelated
- distributed over a computer network

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# DATA DELIVERY ALTERNATIVES

- three orthogonal dimensions:
  - delivery modes
    - Pull-only, push-only, hybrid
  - Frequency
    - Periodic, conditional, ad-hoc凭经验的, irregular
  - communication methods
    - Unicast单播, one-to-many

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# PROMISES承诺 OF DDBSS

- Transparent Management of Distributed and Replicated Data
- Reliability Through Distributed Transactions
- Improved Performance
- Easier System Expansion

# PROMISES承诺 OF DDBSS

- Transparent Management of Distributed and Replicated Data
- Reliability Through Distributed Transactions
- Improved Performance
- Easier System Expansion

# PROMISES承诺 OF DDBSS
# – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Transparency refers to separation of the higher-level semantics of a system from lower-level implementation issues. In other words, a transparent system "hides" the implementation details from users.
- The advantage of a fully transparent DBMS is the high level of support that it provides for the development of complex applications.
- It is obvious that we would like to make all DBMSs (centralized or distributed) fully transparent.

# PROMISES承诺 OF DDBSS
# – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Example:
  - EMP( ENO , ENAME, TITLE)
  - PROJ( PNO , PNAME, BUDGET)
  - SAL( TITLE, AMT )
  - ASG( ENO, PNO , RESP, DUR)

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

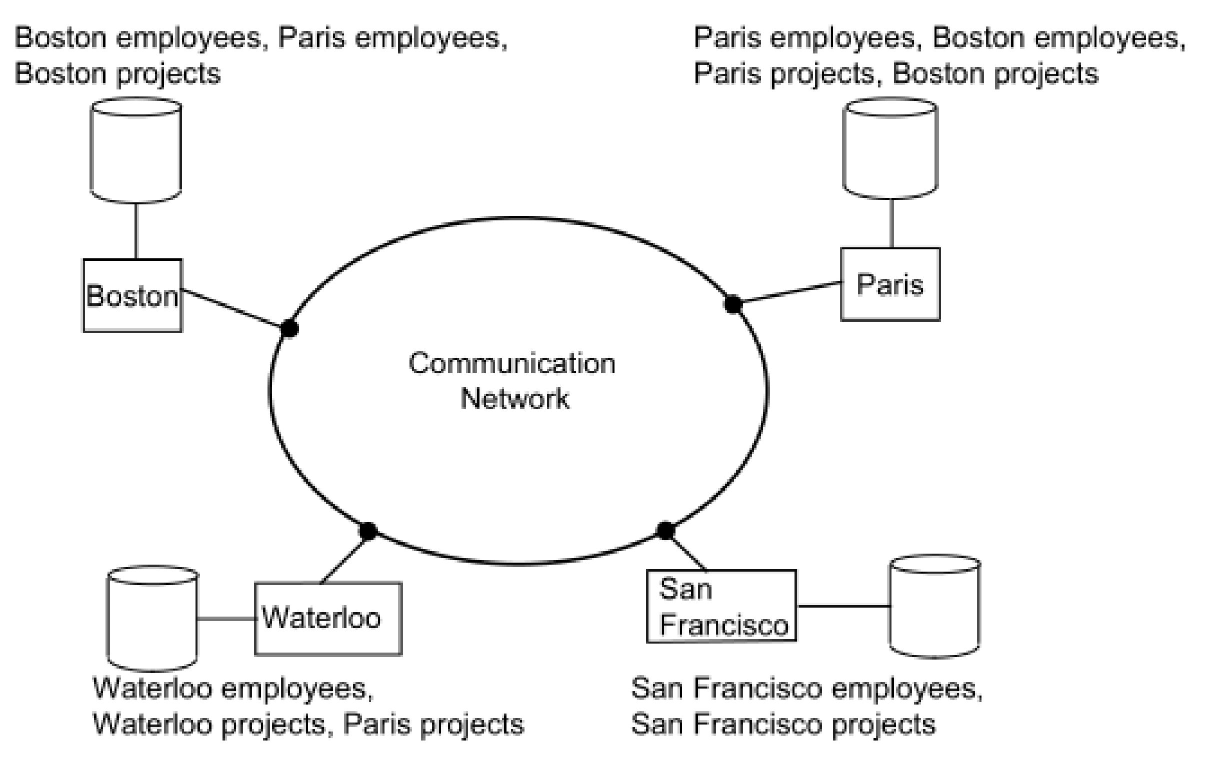
# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

business, performance and reliability

- Example (cont.,):
  - **Fragmentation**分片
    - the employees in Waterloo office are stored in Waterloo, those in the Boston office are stored in Boston, and so forth.

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

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA




performance and reliability

Transparent operations
Under
fragmentation and replication data

**A Distributed Application**

# PROMISES承诺 OF DDBSS
# – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- types of transparencies:*
  - Data Independence
  - Network Transparency
  - Replication Transparency
  - Fragmentation Transparency

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Data Independence
  - Schema definition模式定义
    - Logical data independence逻辑数据独立性
  - Physical data description物理数据描述
    - Physical data independence物理数据独立性

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Network Transparency
  - the user should be protected from the operational details of the network; possibly even hiding the existence of the network.
  - (1)network transparency or distribution transparency
  - (2)location transparency and naming transparency

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Replication Transparency
  - Multi-copies: let us just mention that for performance, reliability, and availability reasons, it is usually desirable to be able to distribute data in a replicated fashion across the machines on a network.
  - From a user's perspective, it is preferable not to be involved with handling copies and having to specify the fact that a certain action can and/or should be taken on multiple copies.

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Fragmentation Transparency
  - it is commonly desirable to divide each database relation into smaller fragments and treat each fragment as a separate database object (i.e., another relation).
  - (1)Horizontal fragmentation水平分片
  - (2)Vertical fragmentation垂直分片
  - a translation from what is called a global query to several fragment queries

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Fragmentation Transparency
  - (1)Horizontal fragmentation水平分片
    - a relation is partitioned into a set of sub-relations each of which have a subset of the tuples (rows) of the original relation

# PROMISES承诺 OF DDBSS
# – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Fragmentation Transparency
  - (2)Vertical fragmentation垂直分片
    - each sub-relation is defined on a subset of the attributes (columns) of the original relation

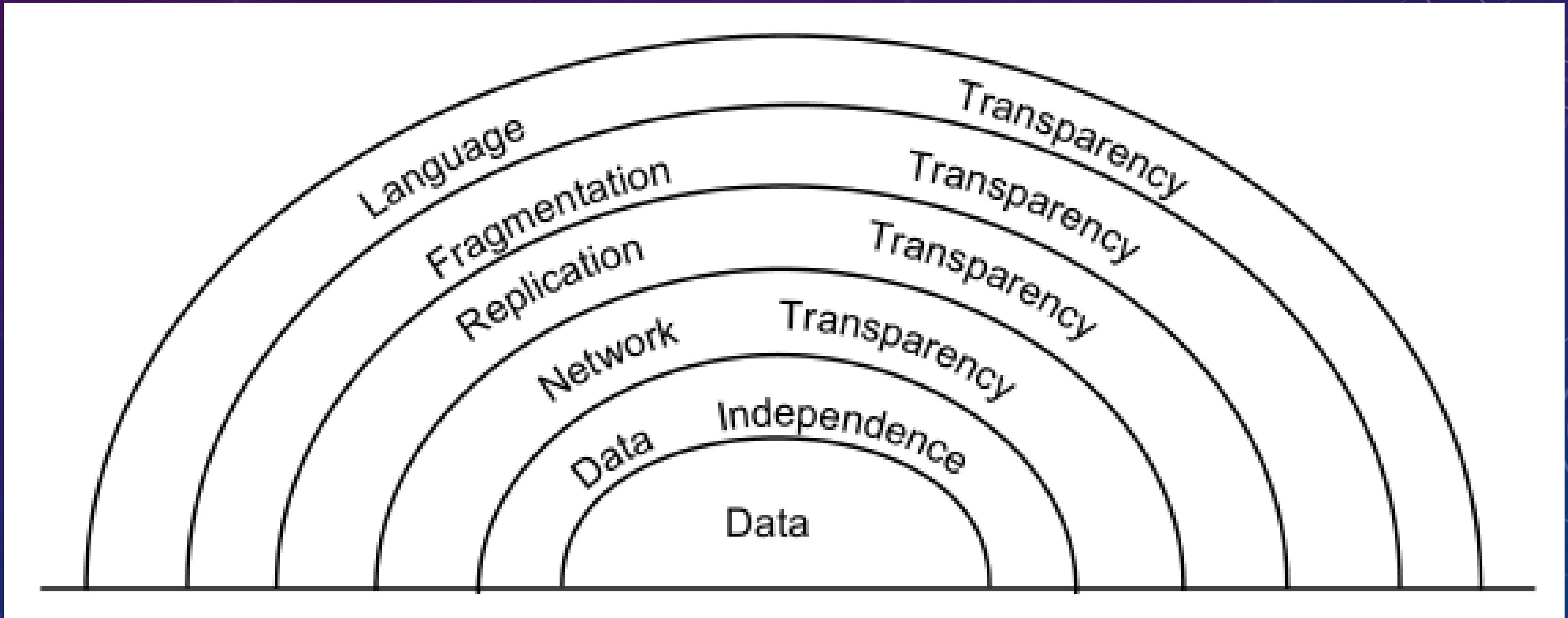
# PROMISES承诺 OF DDBSS
# - TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Who Should Provide Transparency?
  - A full transparency makes the management of distributed data very difficult
  - the level of transparency is inevitably a compromise折衷 between ease of use and the difficulty and overhead cost of providing high levels of transparency

# PROMISES承诺 OF DDBSS – TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA

- Who Should Provide Transparency: three layers
  - (1) We could leave the responsibility of providing transparent access to data resources to the access layer
  - (2) The second layer at which transparency can be provided is the operating system level
  - (3) The third layer at which transparency can be supported is within the DBMS

# PROMISES承诺 OF DDBSS
# - TRANSPARENT MANAGEMENT OF DISTRIBUTED AND REPLICATED DATA



# PROMISES承诺 OF DDBSS

- Transparent Management of Distributed and Replicated Data
- Reliability Through Distributed Transactions
- Improved Performance
- Easier System Expansion

# PROMISES承诺 OF DDBSS – RELIABILITY THROUGH DISTRIBUTED TRANSACTIONS

- single points of failure单点故障
- Transactions and transaction processing
  - A transaction is a basic unit of consistent and reliable computing, consisting of a sequence of database operations executed as an atomic action

# PROMISES承诺 OF DDBSS – RELIABILITY THROUGH DISTRIBUTED TRANSACTIONS

- Concurrency transparency并发透明
  - It transforms a consistent database state to another consistent database state even when a number of such transactions are executed concurrently, and even when failures occur (also called failure atomicity)
- Failure atomicity故障原子性

# PROMISES承诺 OF DDBSS

- Transparent Management of Distributed and Replicated Data
- Reliability Through Distributed Transactions
- Improved Performance
- Easier System Expansion

# PROMISES承诺 OF DDBSS – IMPROVED PERFORMANCE


- (1) data localization数据本地化
  - a distributed DBMS fragments the conceptual database, enabling data to be stored in close proximity to its points of use
- advantages:
  - Since each site handles only a portion of the database, contention for CPU and I/O services is not as severe as for centralized databases.
  - Localization reduces remote access delays that are usually involved in wide area networks

# PROMISES承诺 OF DDBSS – IMPROVED PERFORMANCE

- (2) the inherent parallelism
  - the inherent parallelism of distributed systems may be exploited for inter-query and intra-query parallelism
  - it results from the ability to execute multiple queries at the same time while intra-query parallelism is achieved by breaking up a single query into a number of sub queries each of which is executed at a different site, accessing a different part of the distributed database.

# PROMISES承诺 OF DDBSS

- Transparent Management of Distributed and Replicated Data
- Reliability Through Distributed Transactions
- Improved Performance
- Easier System Expansion

# PROMISES承诺 OF DDBSS
# – EASIER SYSTEM EXPANSION

- In a distributed environment, it is much easier to accommodate increasing database sizes
- Grosh's law
  - it was commonly believed that it would be possible to purchase a fourfold powerful computer if one spent twice as much

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# COMPLICATIONS INTRODUCED BY DISTRIBUTION

- (1)data may be replicated in a distributed environment.
  - A distributed database can be designed so that the entire database, or portions of it, reside at different sites of a computer network.

# COMPLICATIONS INTRODUCED BY DISTRIBUTION

- (2)effect and durability
  - if some sites fail (e.g., by either hardware or software malfunction),
  - or if some communication links fail (making some of the sites unreachable) while an update is being executed,
  - the system must make sure that the effects will be reflected on the data residing at the failing or unreachable sites as soon as the system can recover from the failure.

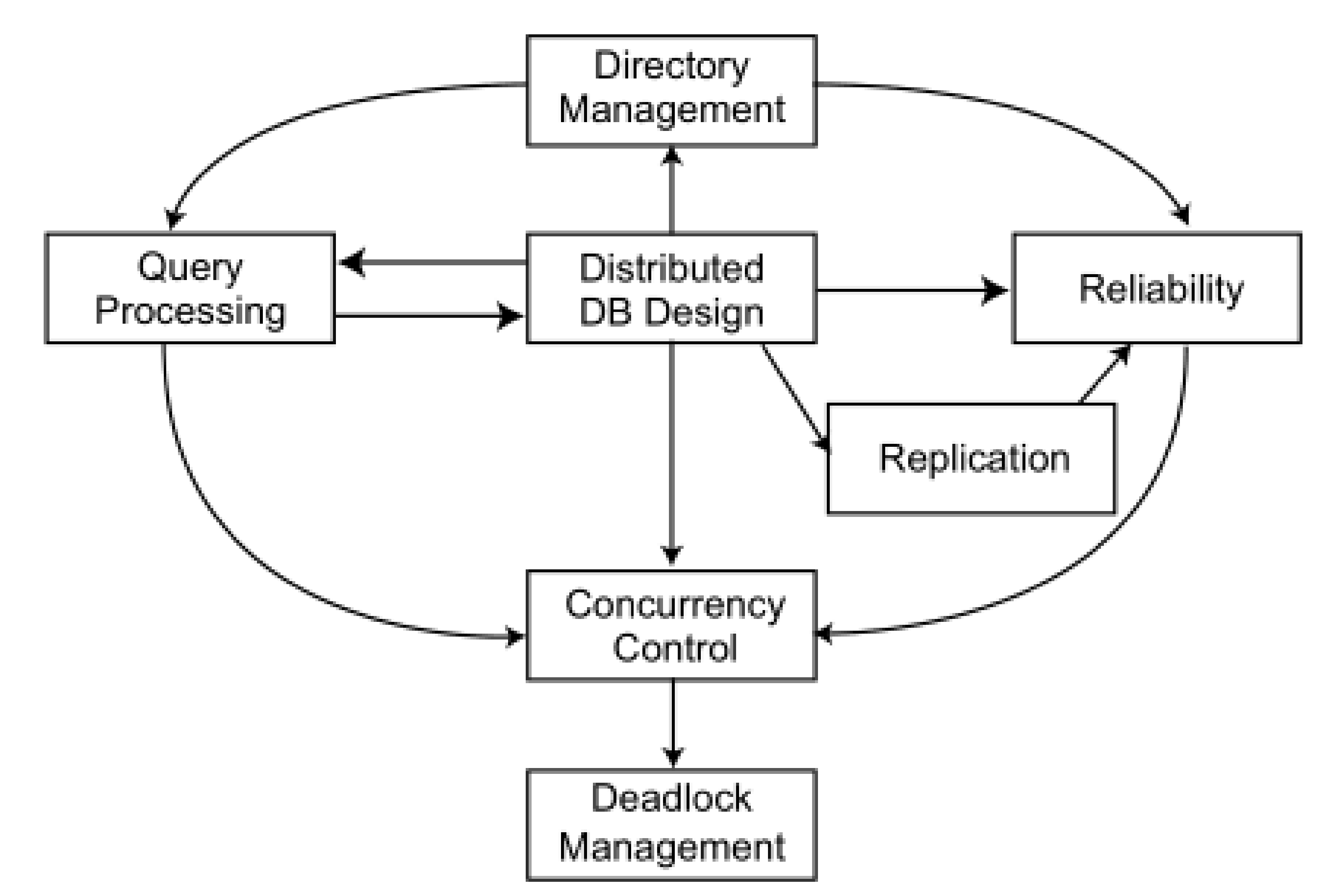
# COMPLICATIONS INTRODUCED BY DISTRIBUTION

- (3) distributed transactions and synchronization
  - since each site cannot have instantaneous information on the actions currently being carried out at the other sites,
  - the synchronization of transactions on multiple sites is considerably harder than for a centralized system.

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# DESIGN ISSUES




Relationship Among Research Issues

# DESIGN ISSUES

- (1) Distributed Database Design
- two basic alternatives to placing data
  - (a) partitioned (or non-replicated)划分
  - (b) replicated复制
    - Fully replicated, partially replicated

# DESIGN ISSUES

- (1) Distributed Database Design
- two fundamental design issues:
  - Fragmentation(分片,划分): the separation of the database into partitions called fragments片段
  - Distribution(分配,分布): the optimum distribution of fragments
- The research in this area mostly involves mathematical programming in order to minimize the combined cost of storing the database, processing transactions against it, and message communication among sites

# DESIGN ISSUES

- (2) 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 DDBS or local to each site;
  - it can be centralized at one site or distributed over several sites; there can be a single copy or multiple copies.

# DESIGN ISSUES

- (3) Distributed Query Processing
  - Query processing deals with designing algorithms that analyze queries and convert them into a series of data manipulation operations.
  - The problem is how to decide on a strategy for executing each query over the network in the most cost-effective way, however cost is defined.
  - The factors to be considered are the distribution of data, communication costs, and lack of sufficient locally-available information.
  - The objective is to optimize where the inherent parallelism is used to improve the performance of executing the transaction, subject to the above-mentioned constraints.
  - NP-hard

# DESIGN ISSUES

- (4) Distributed Concurrency Control并发控制
  - Concurrency control involves the synchronization of accesses to the distributed database, such that the integrity完整性 of the database is maintained.
  - It is one of the most extensively studied problems in the DDBS field
  - This problem in a distributed context is somewhat different than in a centralized framework. One not only has to worry about the integrity of a single database, but also about the consistency of multiple copies of the database
  - mutual consistency相互一致性:The condition that requires all the values of multiple copies of every data item to converge to the same value

# DESIGN ISSUES

- (4) Distributed Concurrency Control (cont.,)
- mutual consistency相互一致性:
  - The condition that requires all the values of multiple copies of every data item to converge to the same value
  - (a) pessimistic悲观方法:synchronizing the execution of user requests before the execution starts (locking method)
  - (b) optimistic乐观方法:executing the requests and then checking if the execution has compromised the consistency of the database (timestamping method)

# DESIGN ISSUES

- (5) Distributed Deadlock Management
  - Deadlock: The competition among users for access to a set of resources can result in a deadlock if the synchronization mechanism is based on locking
  - prevention, avoidance, and detection/recovery
  - Similar to OS deadlock

# DESIGN ISSUES

- (6) Reliability of Distributed DBMS
  - one of the potential advantages of distributed systems is improved reliability and availability.
  - however, is not a feature that comes automatically.
  - It is important that mechanisms be provided to ensure the consistency of the database as well as to detect failures and recover from them

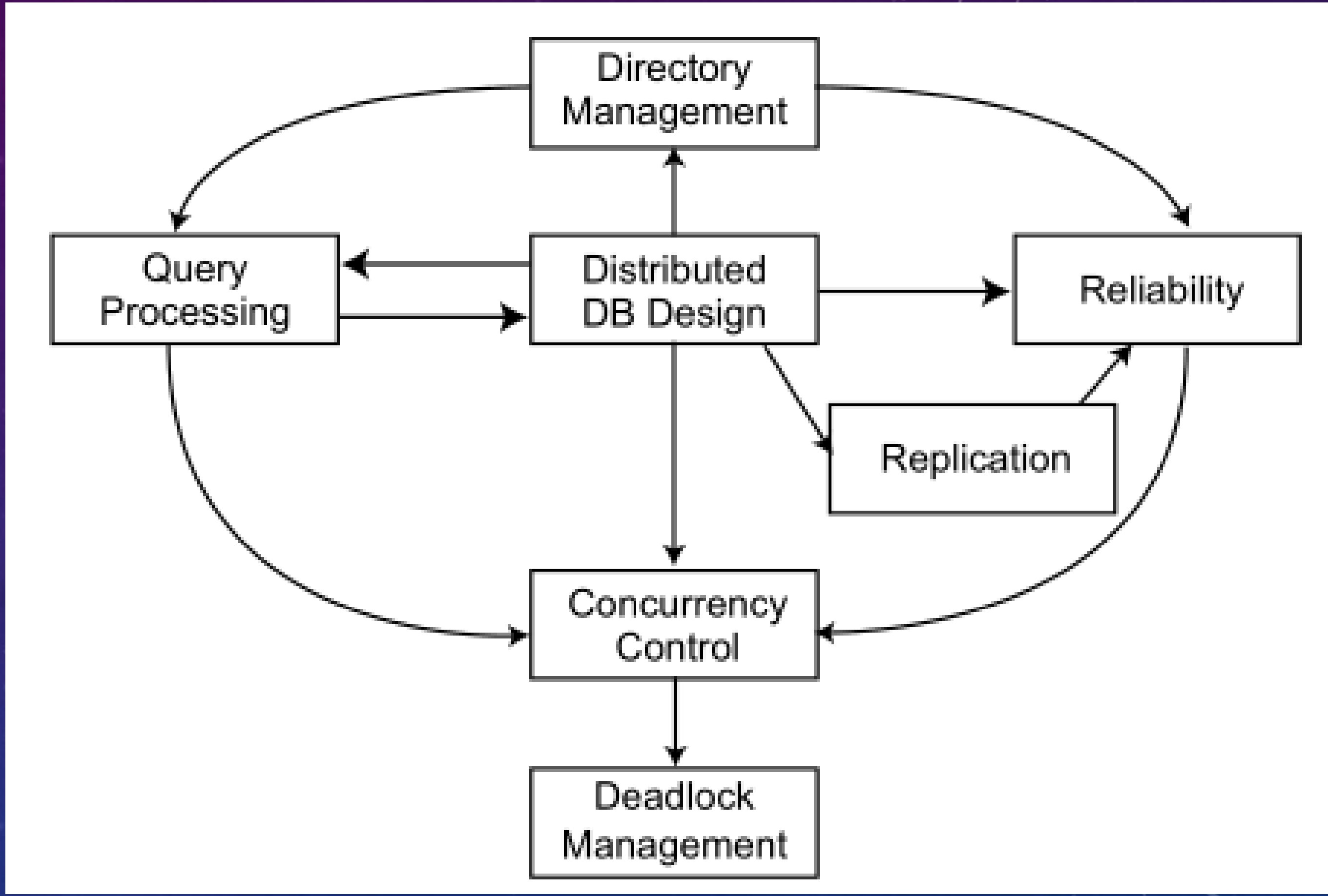
# DESIGN ISSUES

- (7) Replication
  - If the distributed database is (partially or fully)replicated, itis necessary to implement protocols that ensure the consistency of the replicas, i.e., copies of the same data item have the same value.
  - (a)These protocols can be eager及时 in that they force the updates to be applied to all the replicas before the transaction completes
  - (b) they may be lazy惰性 so that the transaction updates one copy (called the master主拷贝) from which updates are propagated传播 to the others after the transaction completes.

# DESIGN ISSUES
# – ADDITIONAL ISSUES

- Multidatabase systems
  - federated databases联邦数据库
  - data integration systems数据集成系统
- P2P DB
- Web DB

# DESIGN ISSUES – RELATIONSHIP AMONG PROBLEMS




Relationship Among Research Issues

# OBJECTIVES

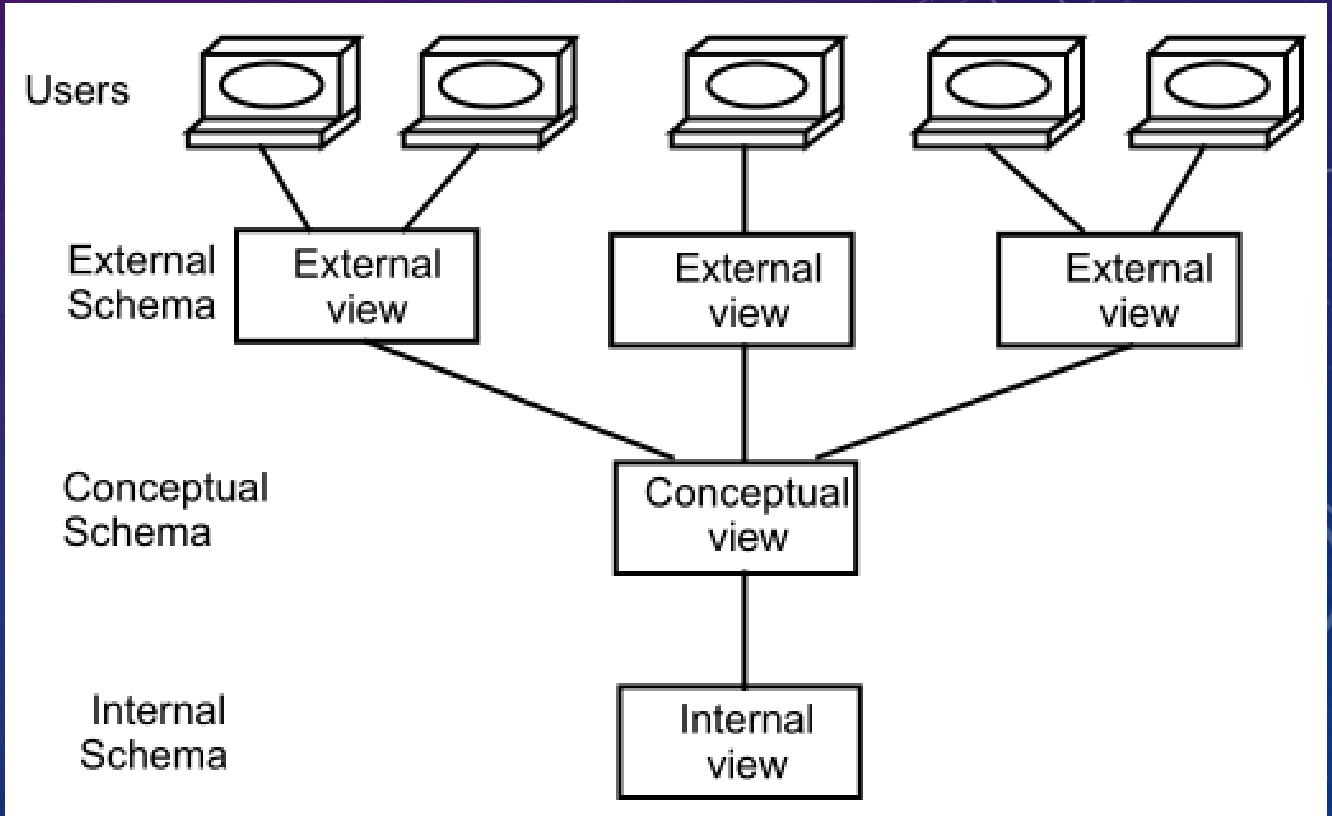
- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# DISTRIBUTED DBMS ARCHITECTURE

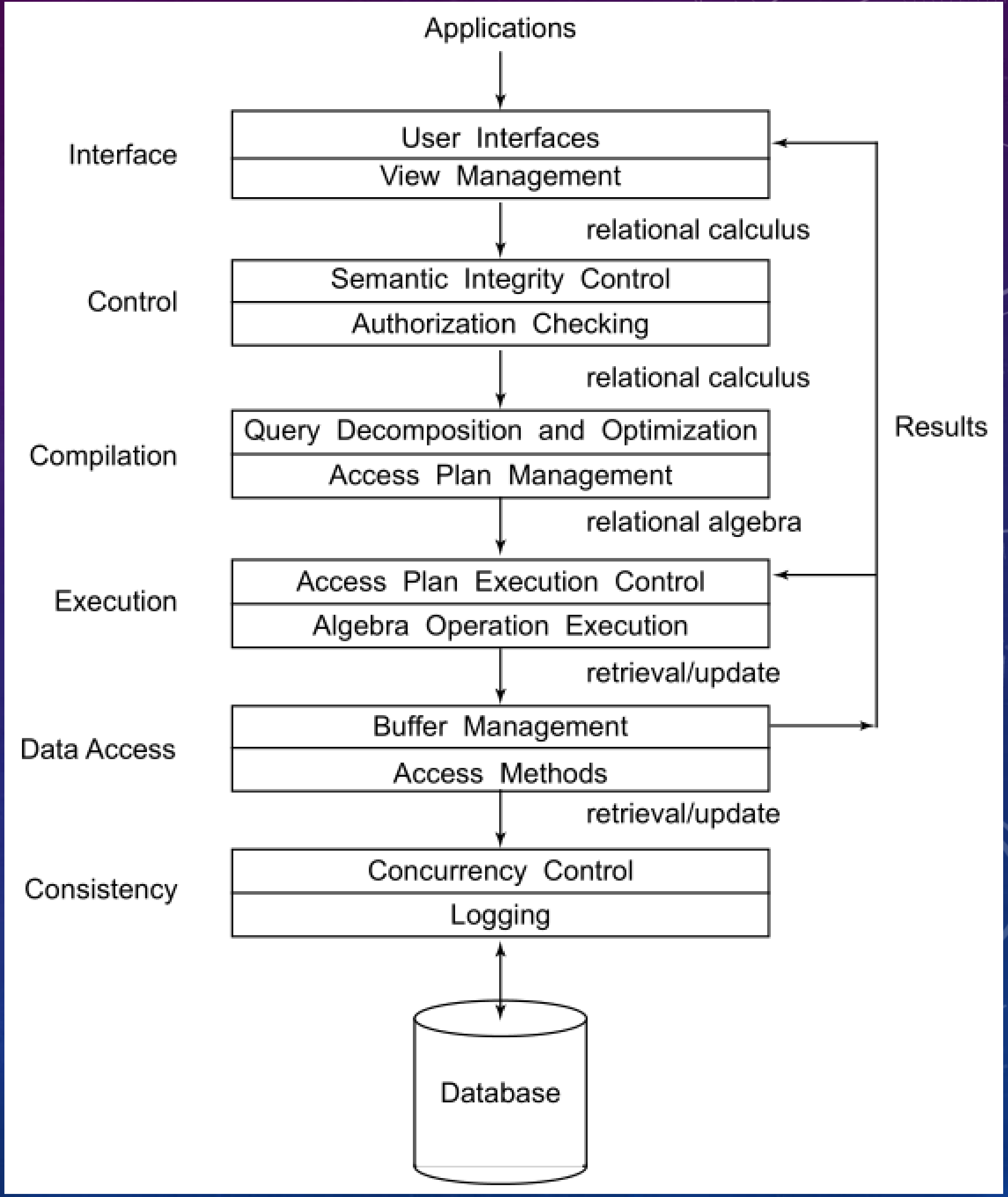
- The architecture of a system defines its structure.
  - This means that the components of the system are identified, the function of each component is specified, and the interrelationships and interactions among these components are defined.
- three “reference” architectures:
  - client/server systems
  - peer-to-peer distributed DBMS
  - multidatabase systems

# DISTRIBUTED DBMS ARCHITECTURE – ANSI/SPARC ARCHITECTURE

- The ANSI/SPARC Architecture, 1975
  - Internal Schema
  - Conceptual Schema
  - External Schema
- Data independence
  - Logical data independence
  - Physical data independence

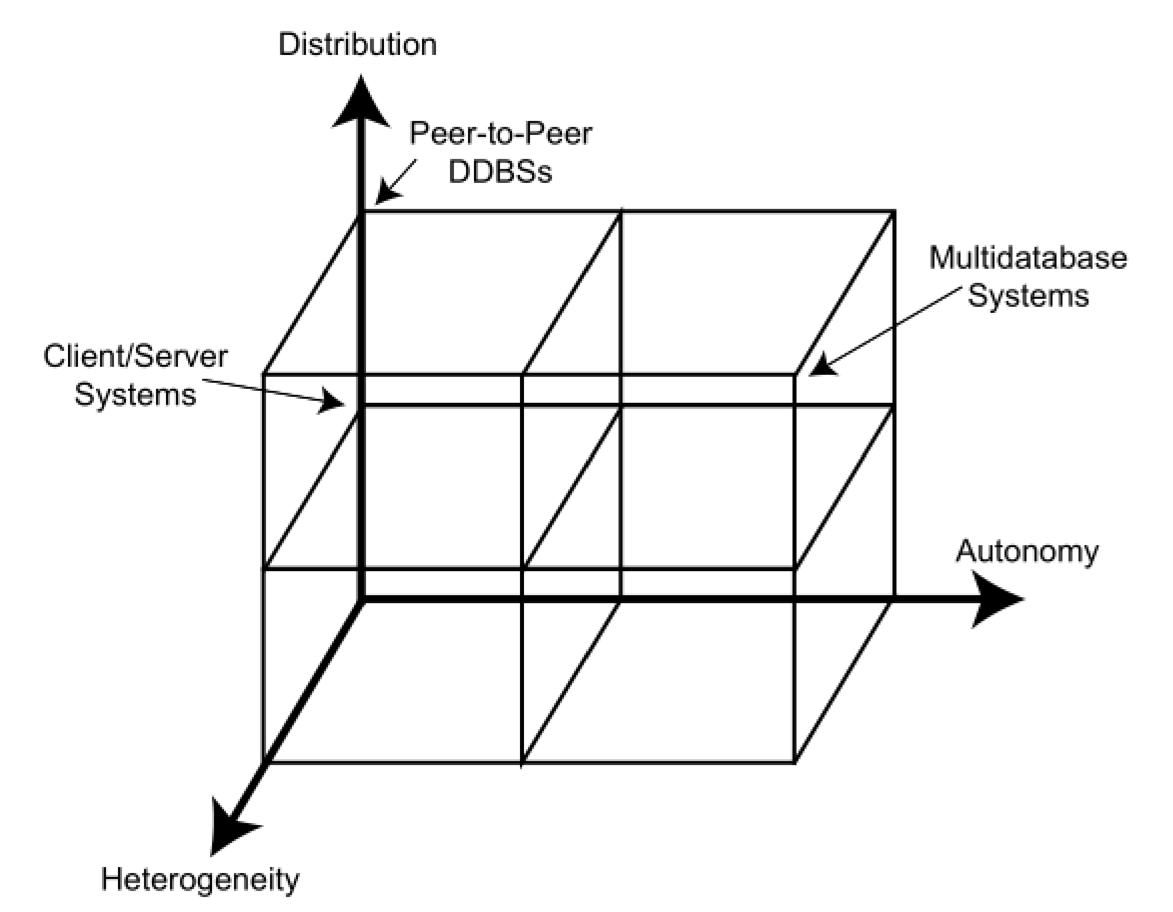


# DISTRIBUTED DBMS ARCHITECTURE – A GENERIC CENTRALIZED DBMS ARCHITECTURE



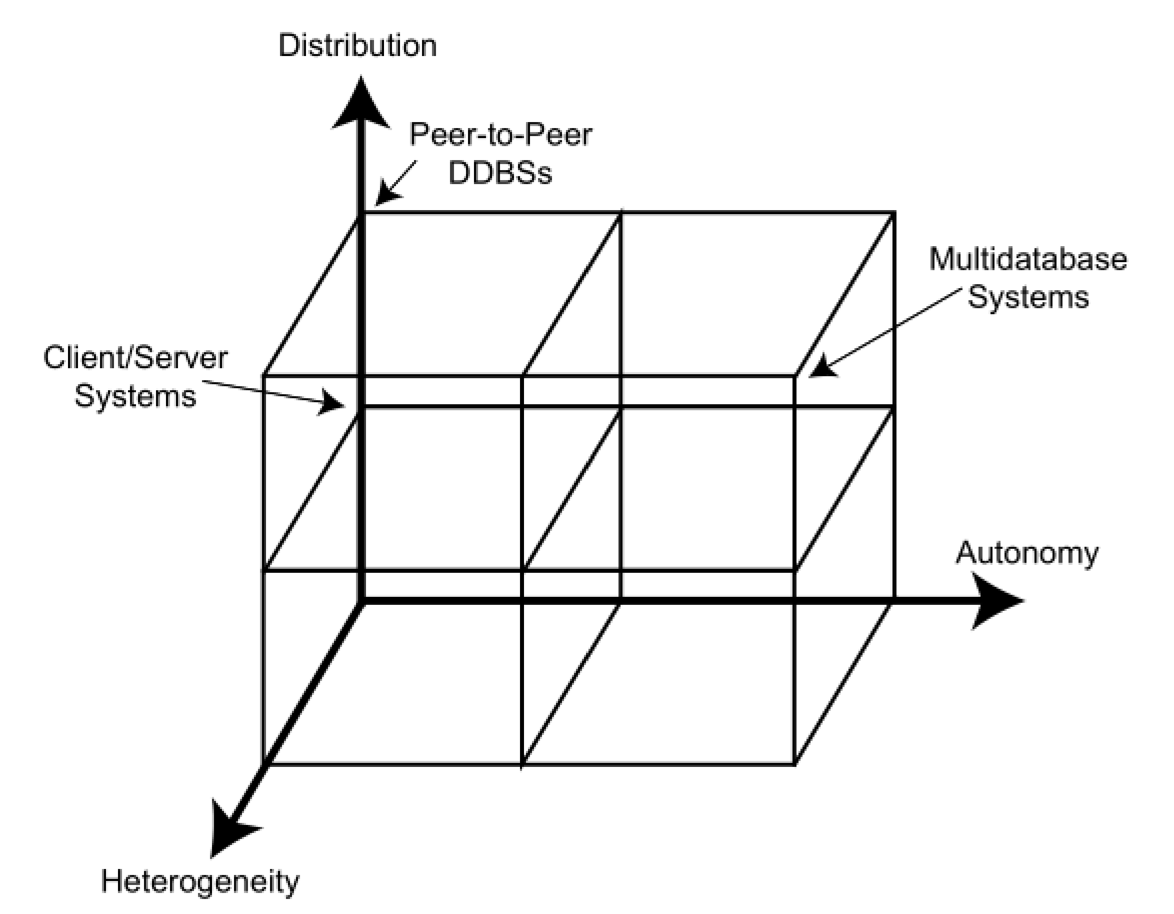
# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- classification：
  - (1) the autonomy of local systems
  - (2) their distribution
  - (3) their heterogeneity



# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- classification：
  - (1) the autonomy of local systems
  - (2) their distribution
  - (3) their heterogeneity



# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- Autonomy自治性
  - Autonomy, in this context, refers to the distribution of control, not of data.
  - It indicates the degree to which individual DBMSs can operate independently

# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- Autonomy自治性(cont.,)
- Requirements of an autonomous system[Gligor and PopescuZeletin,1986]:
  - (1)The local operations of the individual DBMSs are not affected by their participation in the distributed system.
  - (2)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.
  - (3)System consistency or operation should not be compromised妥协 when individual DBMSs join or leave the distributed system.

# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

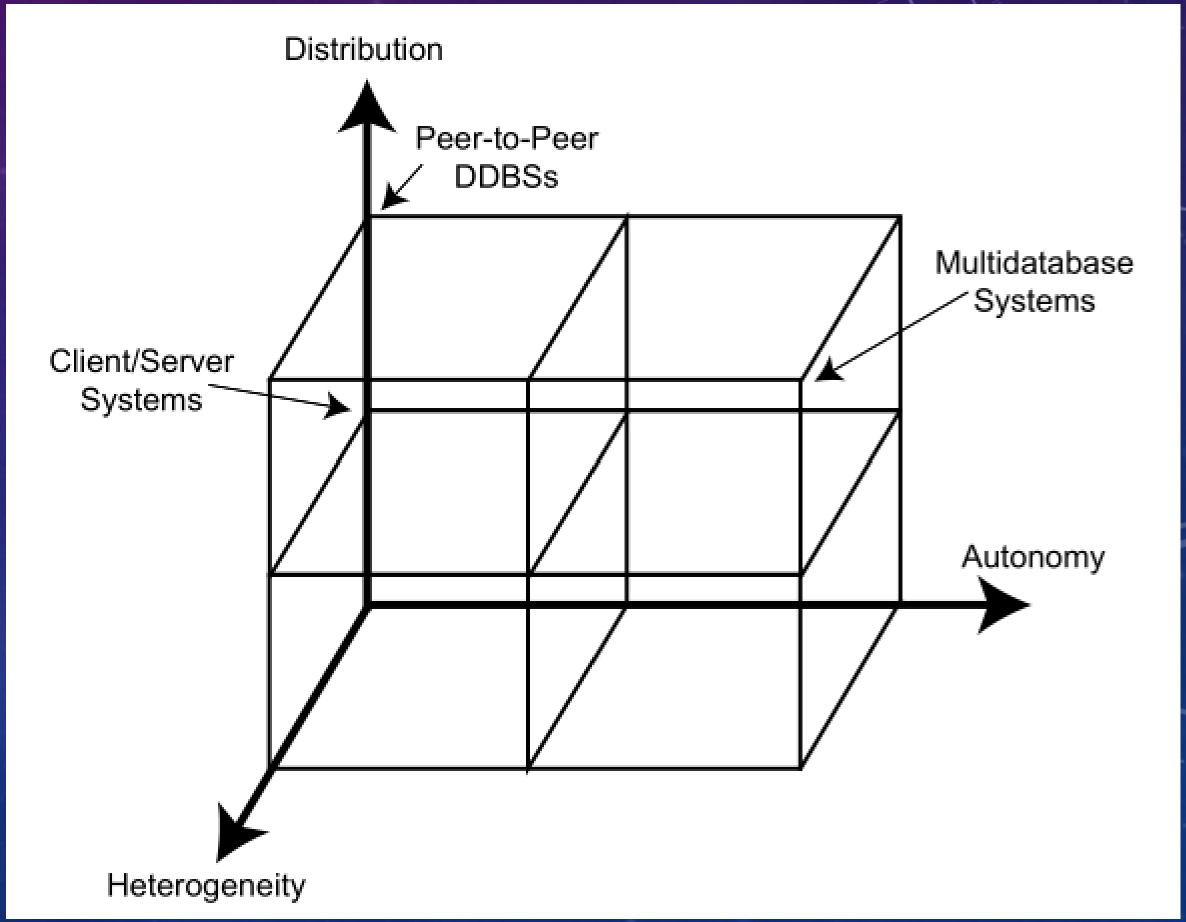
- Autonomy自治性(cont.,)
- the dimensions of autonomy [Du and Elmagarmid, 1989]:
  - 1. Design autonomy: Individual DBMSs are free to use the data models and transaction management techniques that they prefer.
  - 2. Communication autonomy: Each of the individual DBMSs is free to make its own decision as to what type of information it wants to provide to the other DBMSs or to the software that controls their global execution.
  - 3. Execution autonomy: Each DBMS can execute the transactions that are submitted to it in any way that it wants to.

# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- Autonomy自治性(cont.,)
- the types of DDBS based on different autonomy
  - tight integration systems紧密集成的系统
    - 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.
  - semiautonomous systems半自治系统
    - it consist of DBMSs that can (and usually do) operate independently, but have decided to participate in a federation to make their local data sharable.
  - total isolation systems全孤立系统
    - the individual systems are stand-alone DBMSs that know neither of the existence of other DBMSs nor how to communicate with them.

# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- classification：
  - (1) the autonomy of local systems
  - (2) their distribution
  - (3) their heterogeneity

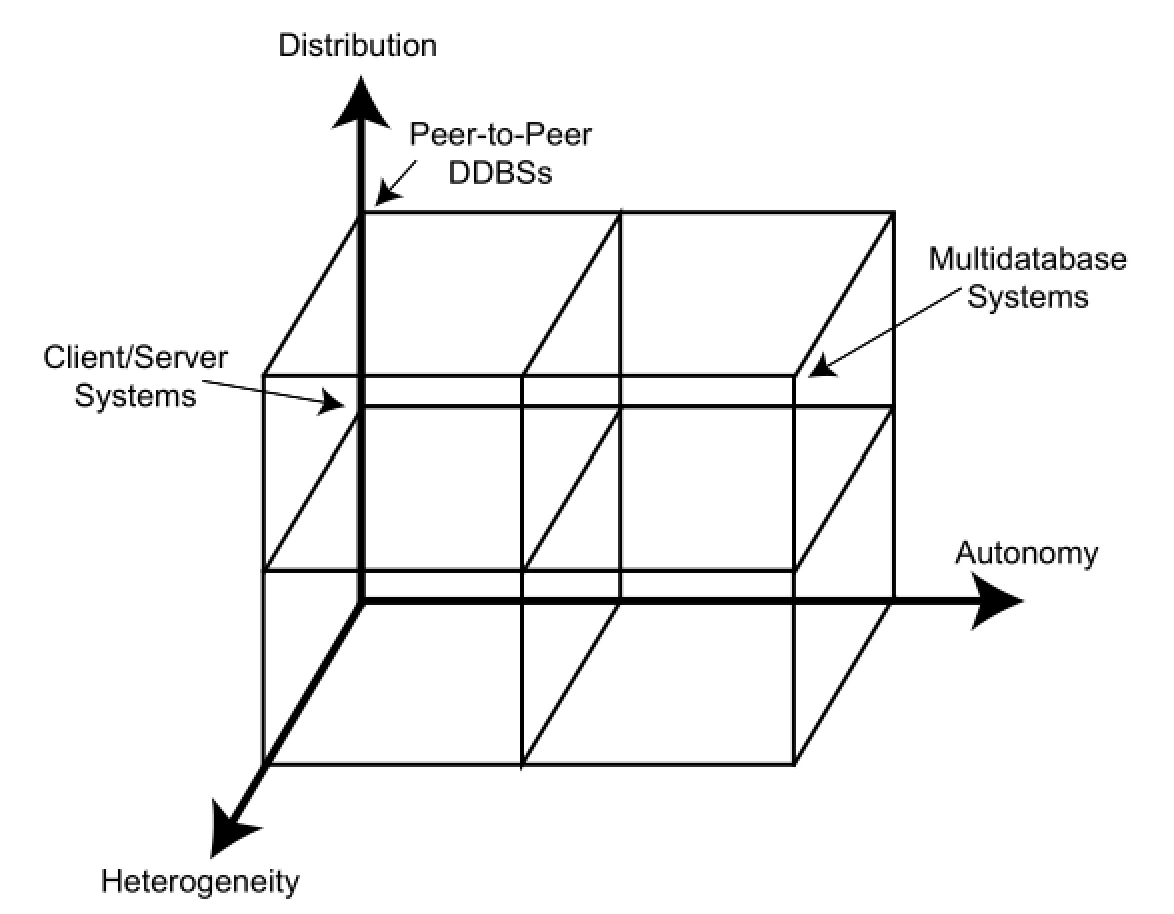


# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- Distribution分布
  - the physical distribution of data over multiple sites
- taxonomy
  - (1) client/server distribution
  - (2) peer-to-peer distribution (or full distribution)
  - (3) non-distributed

# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- classification：
  - (1) the autonomy of local systems
  - (2) their distribution
  - (3) their heterogeneity



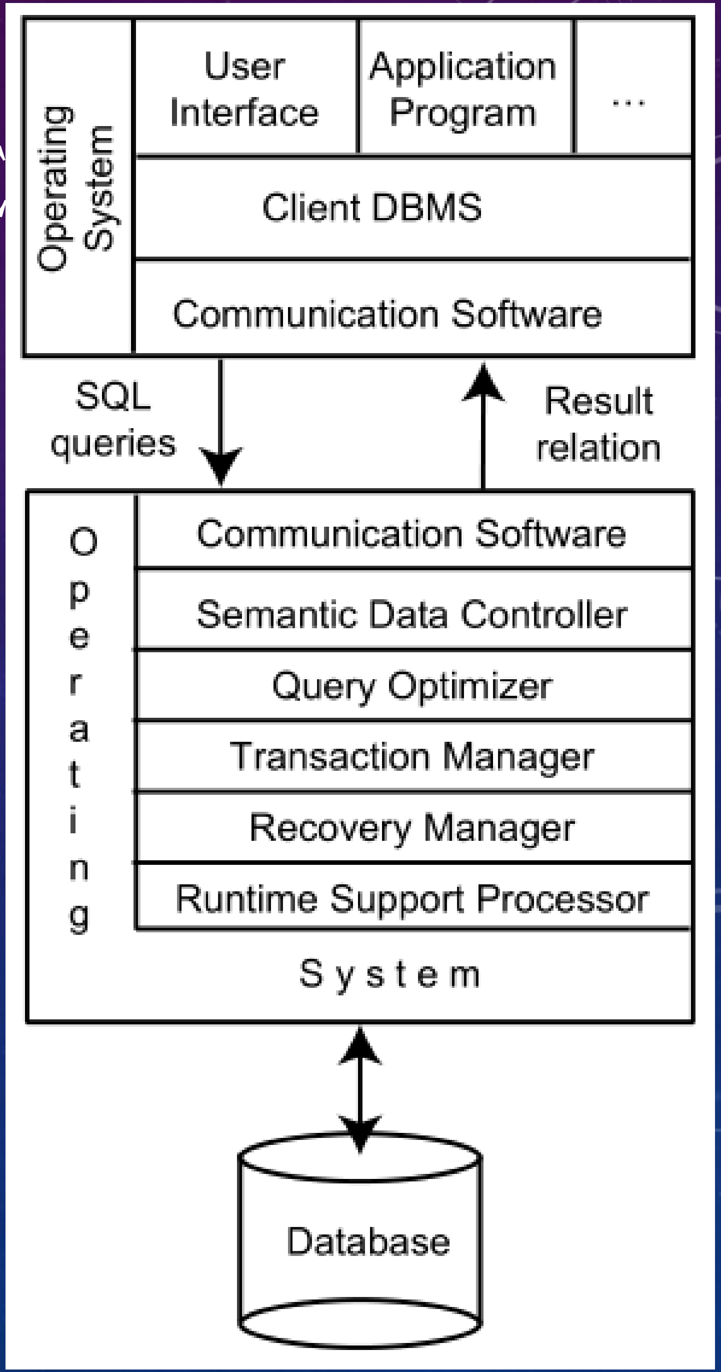
# DISTRIBUTED DBMS ARCHITECTURE – ARCHITECTURAL MODELS FOR DISTRIBUTED DBMSS

- Heterogeneity异构性
  - Heterogeneity may occur in various forms in distributed systems, ranging from hardware heterogeneity and differences in networking protocols to variations in data managers
  - data models, query languages, and transaction management protocols

# DISTRIBUTED DBMS ARCHITECTURAL ALTERNATIVES

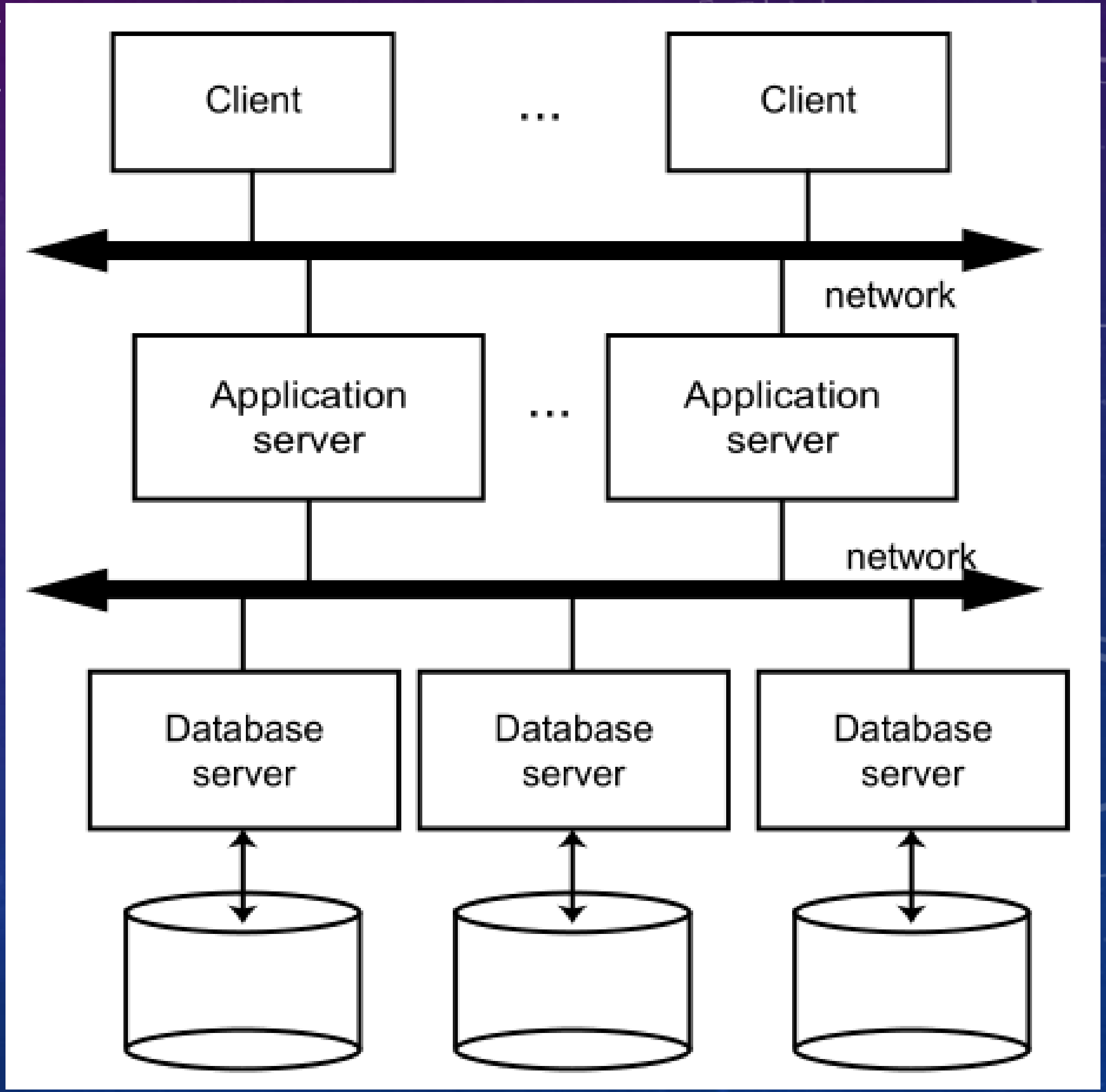
# DISTRIBUTED DBMS ARCHITECTURE – CLIENT/SERVER SYSTEMS

- client/server computing
- client/server DBMS
  - Multiple client/Single server
  - Multiple client/Multiple server



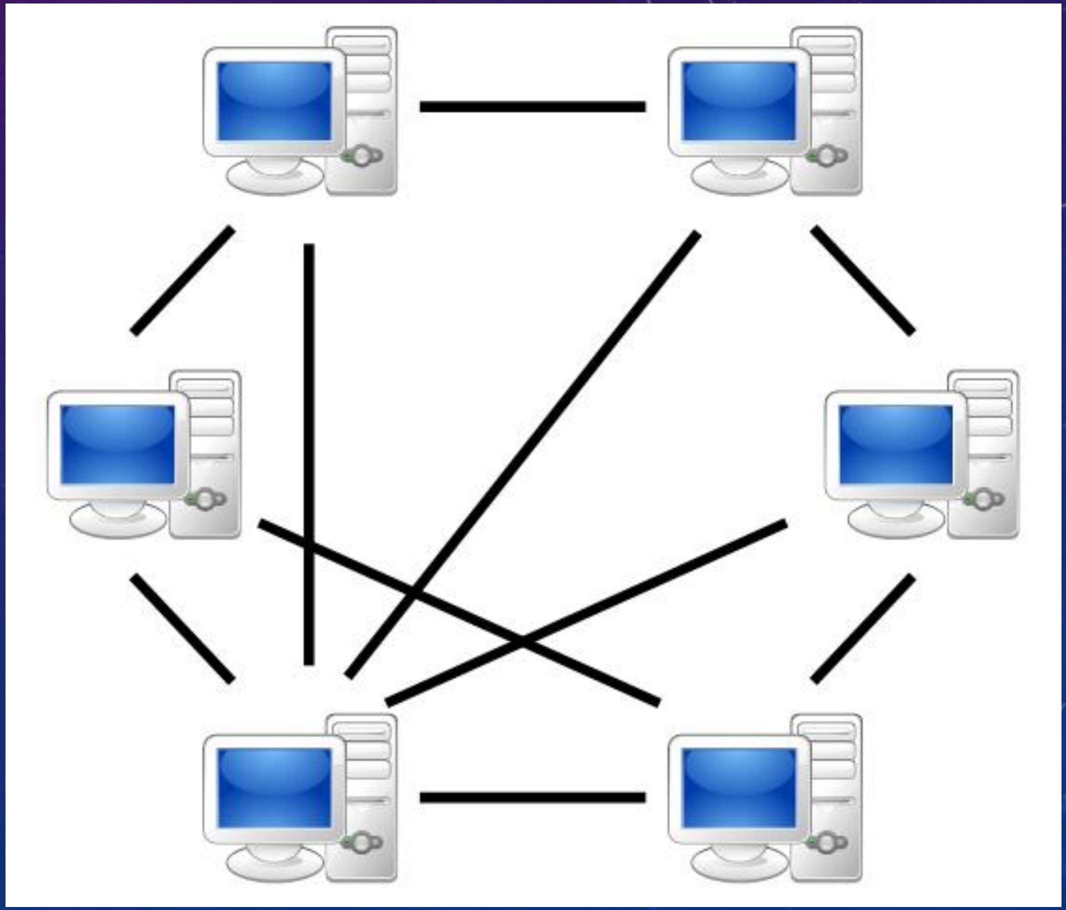
# DISTRIBUTED DBMS ARCHITECTURE – CLIENT/SERVER SYSTEM

- C/S/S
  - Client server (web server)
  - Application server
  - Database server



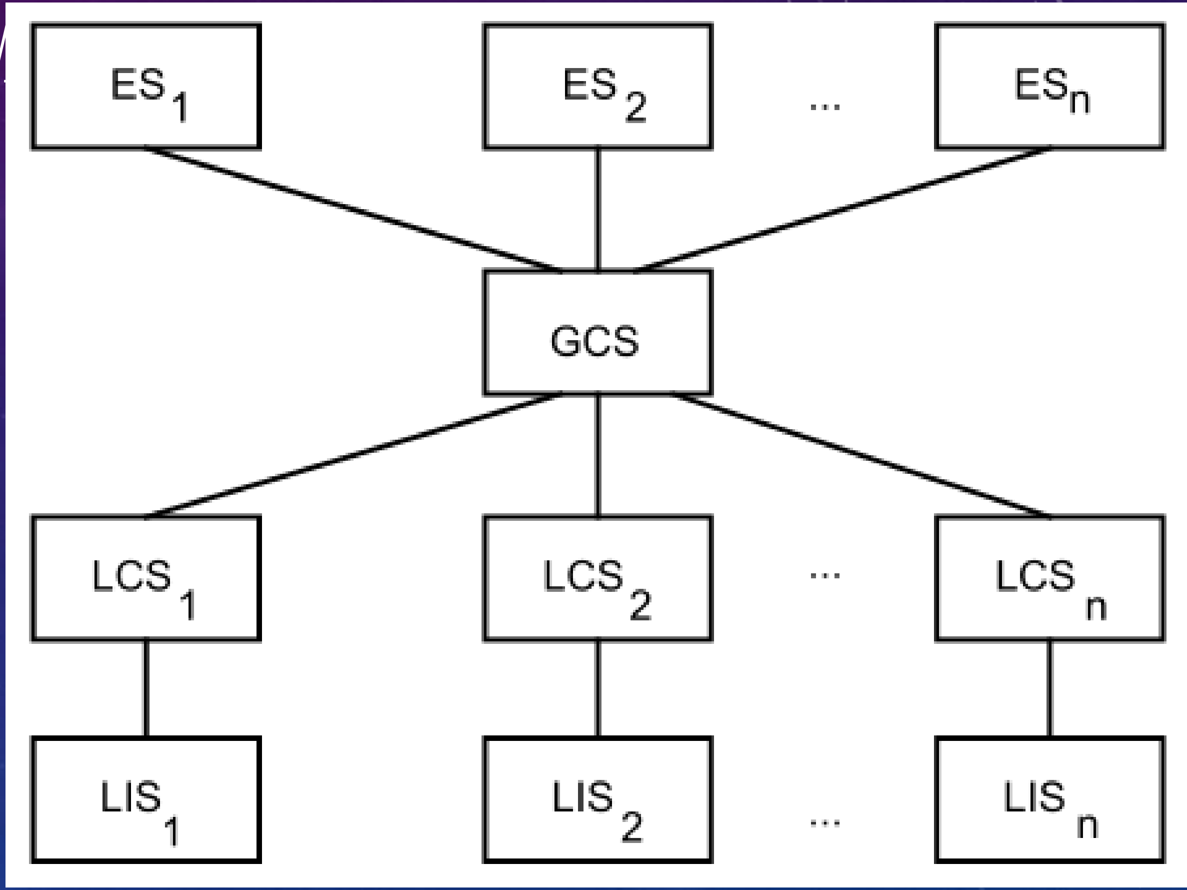
# DISTRIBUTED DBMS ARCHITECTURE – PEER-TO-PEER SYSTEMS

- Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers.
- Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.
- Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.



# DISTRIBUTED DBMS ARCHITECTURE – PEER-TO-PEER SYSTEM

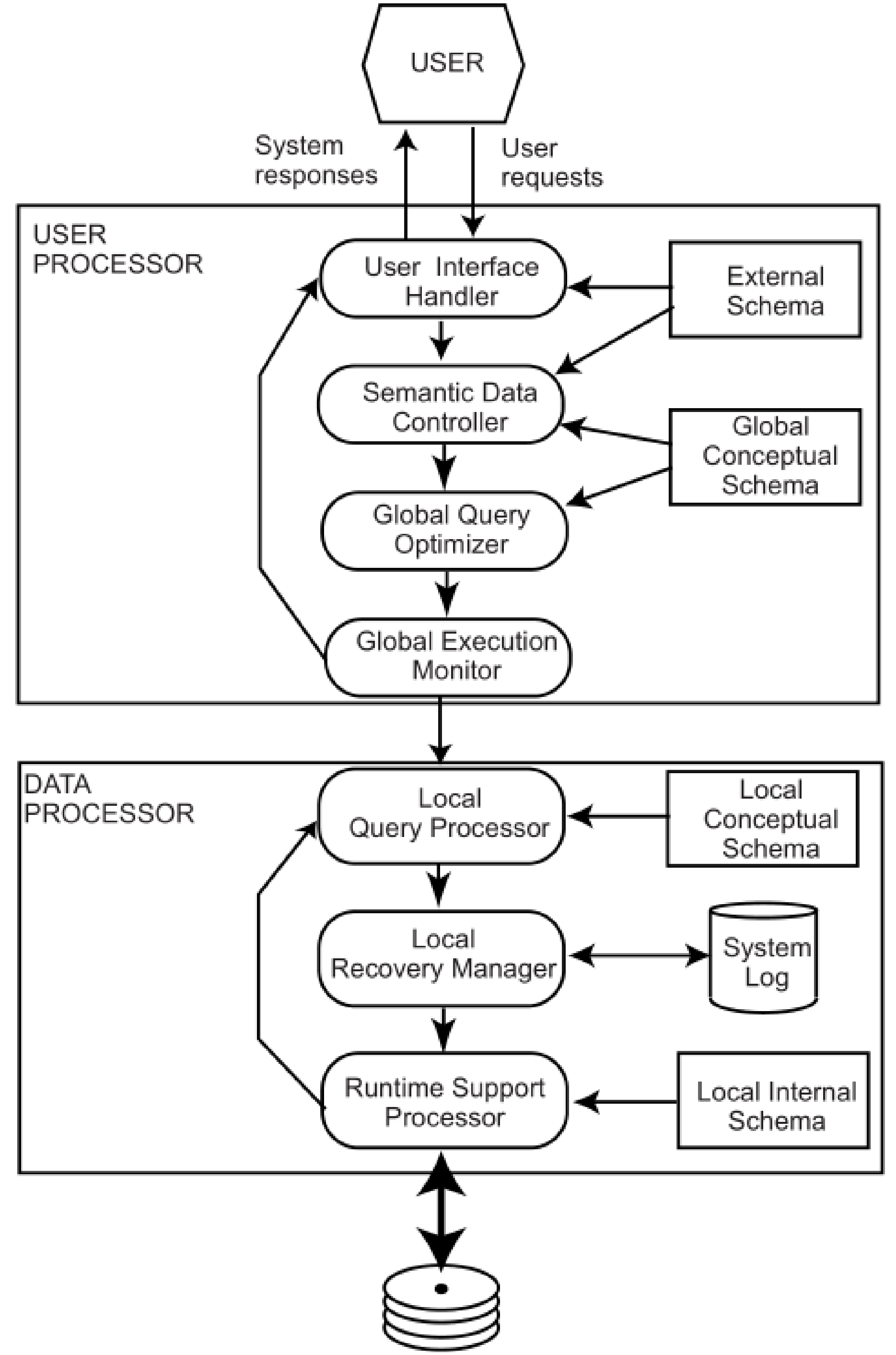
- Local internal schema, LIS 内部模式
- Global conceptual schema, GCS全局概念模式
- External schema, ES外部模式




Distributed Database Reference Architecture

# DISTRIBUTED DBMS ARCHITECTURE – PEER-TO-PEER SYSTEMS

- (1) the user processor
- (2) the data processor

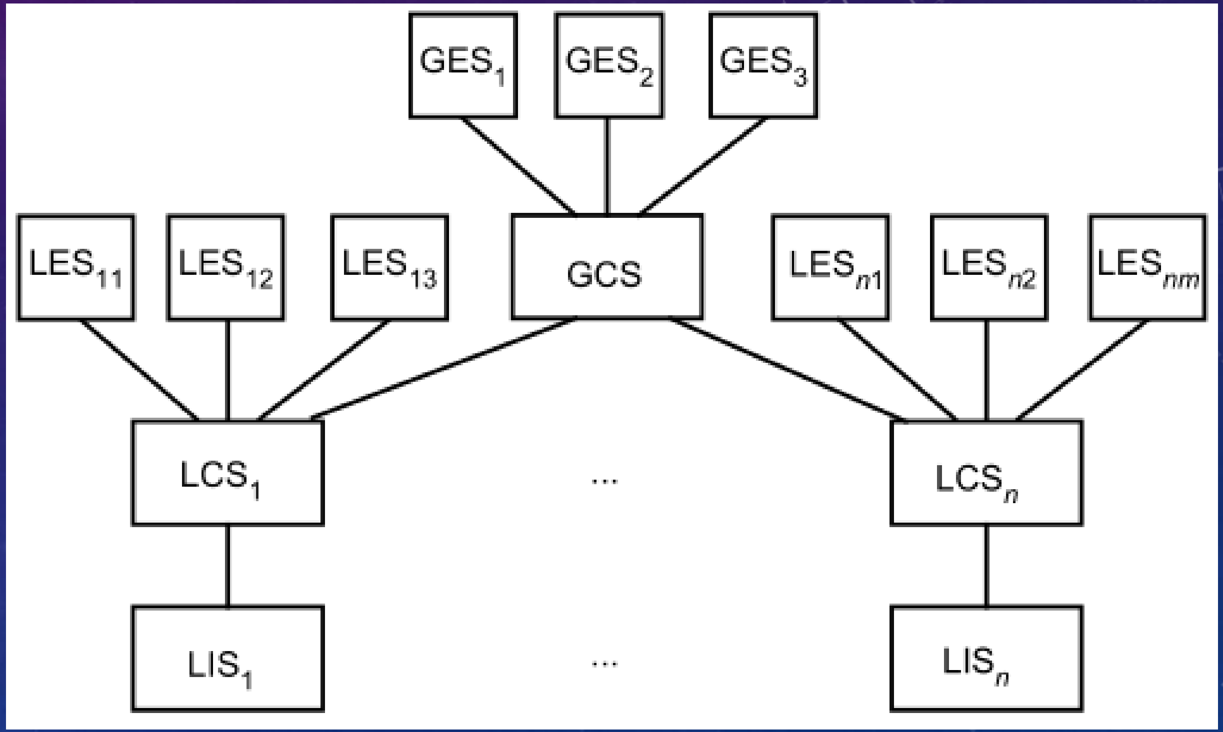


# DISTRIBUTED DBMS ARCHITECTURE – MULTIDATABASE SYSTEM ARCHITECTURE

- Multi-database systems (MDBS)
  - It 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.
- Export schema, ES出口模式

# DISTRIBUTED DBMS ARCHITECTURE – MULTIDATABASE SYSTEM ARCHITECTURE

- (1)the distributed multi-DBMSs
- (2)distributed DBMSs
- The differences in the level of autonomy between the distributed multi-DBMSs and distributed DBMSs are also reflected in their architectural models

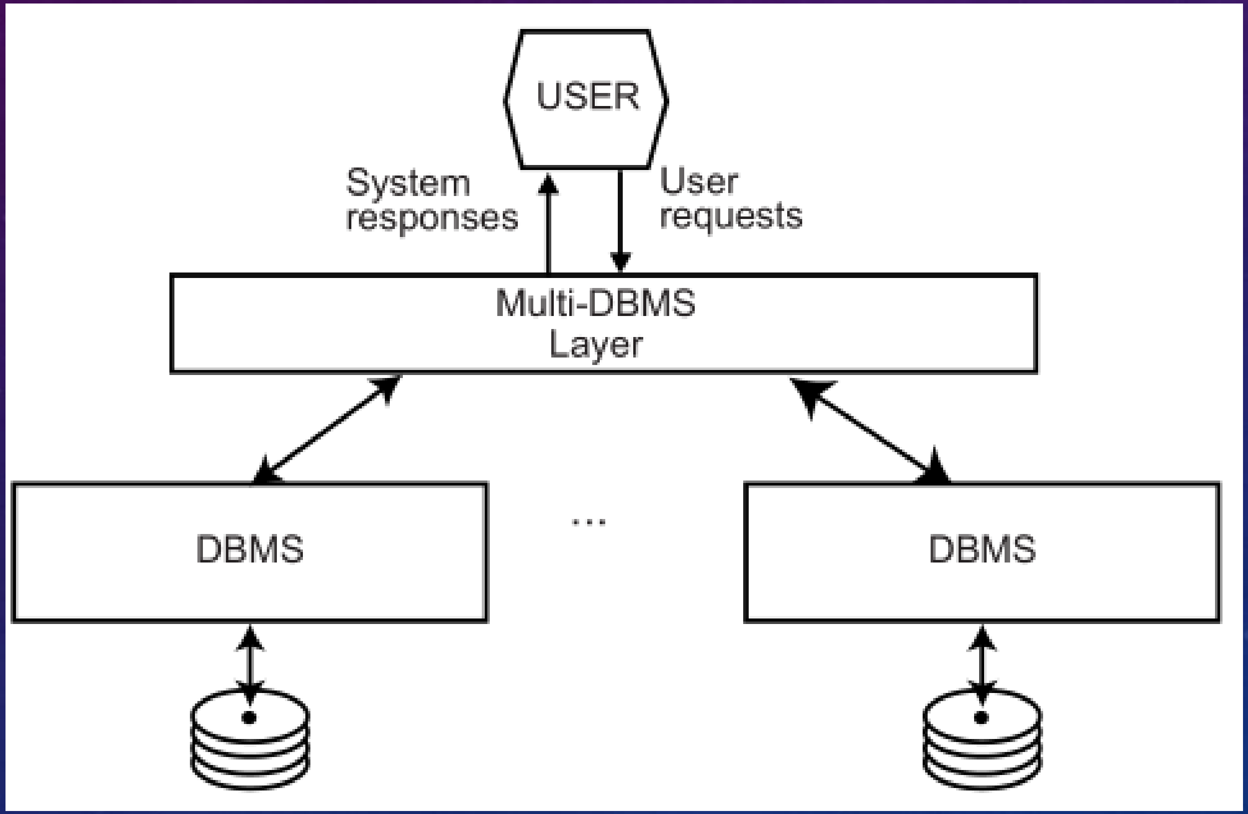



MDBS Architecture with a GCS
Global Conceptual Schema

# DISTRIBUTED DBMS ARCHITECTURE – MULTIDATABASE SYSTEM ARCHITECTURE

- If heterogeneity exists in the system, then two implementation alternatives exist: unilingual单语言 and multilingual多语言
  - A unilingual multi-DBMS requires the users to utilize possibly different data models and languages when both a local database and the global database are accessed.
  - the multilingual architecture permits each user to access the global database (i.e., data from other databases) by means of an external schema, defined using the language of the user's local DBMS.

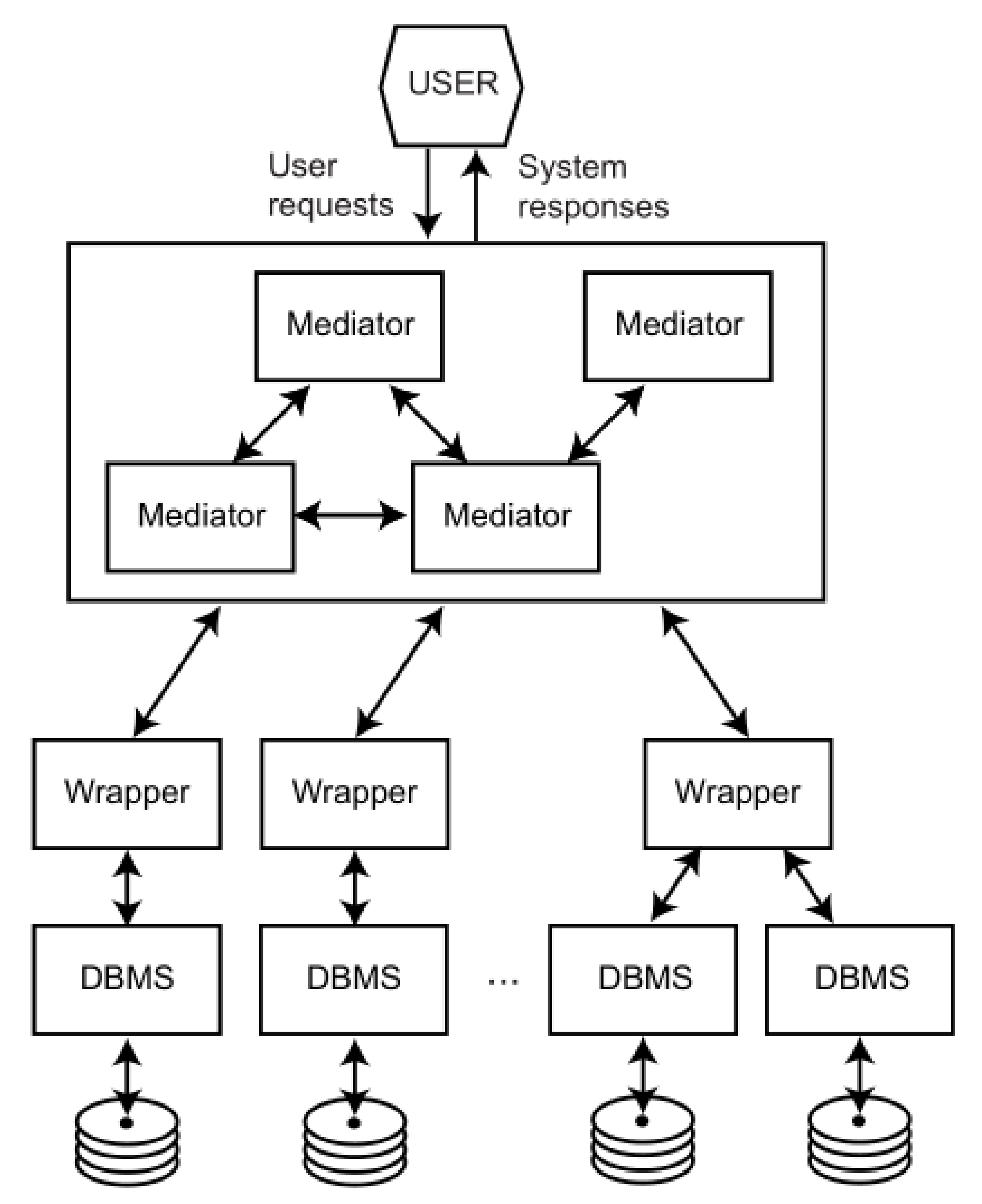
# DISTRIBUTED DBMS ARCHITECTURE – MULTIDATABASE SYSTEM ARCHITECTURE




Components of an MDBS

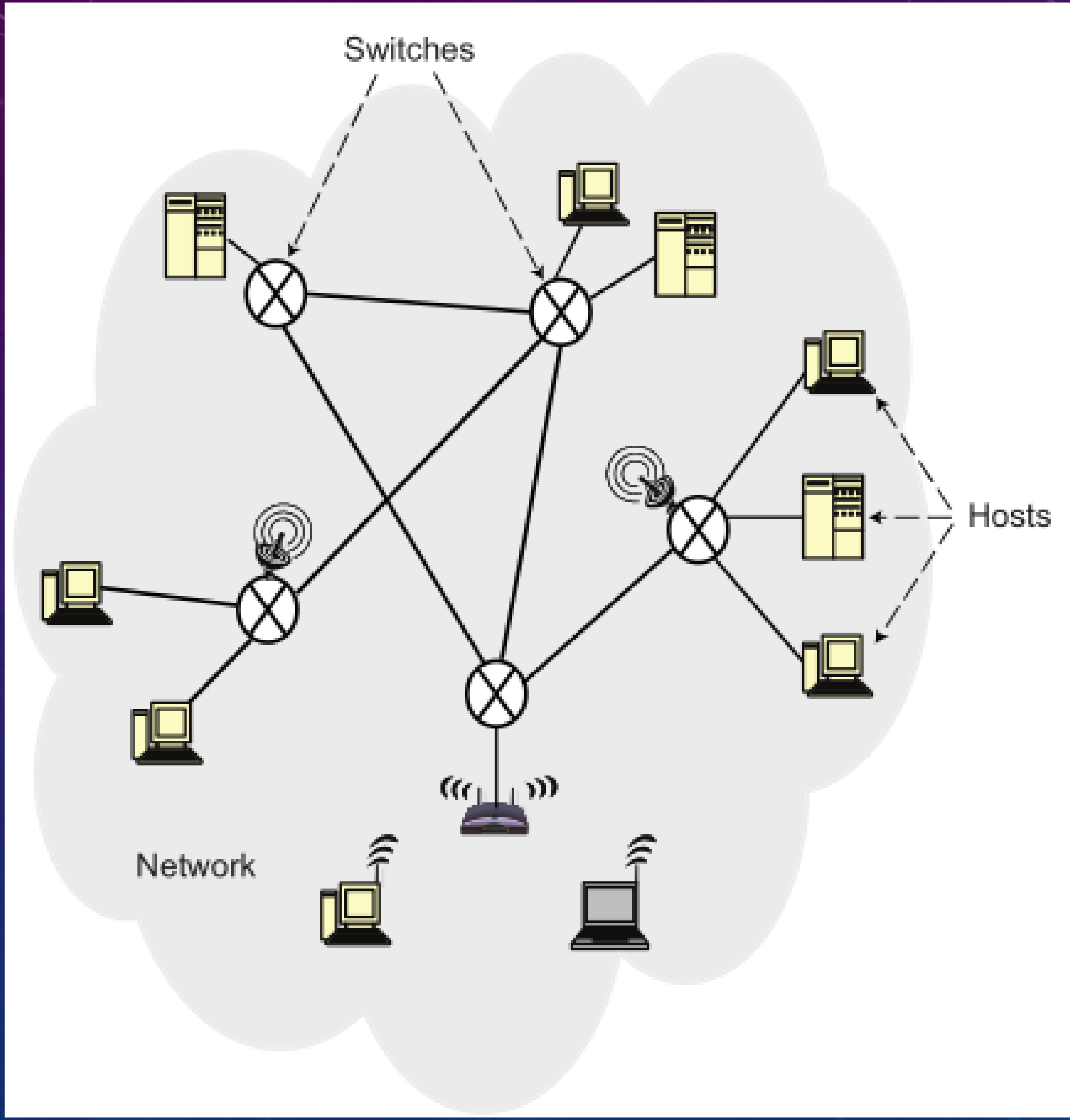
# DISTRIBUTED DBMS ARCHITECTURE – MULTIDATABASE SYSTEM ARCHITECTURE

- Mediator中介程序
  - a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications
- Wrapper包装程序
  - whose task is to provide a mapping between a source DBMSs view and the mediators' view

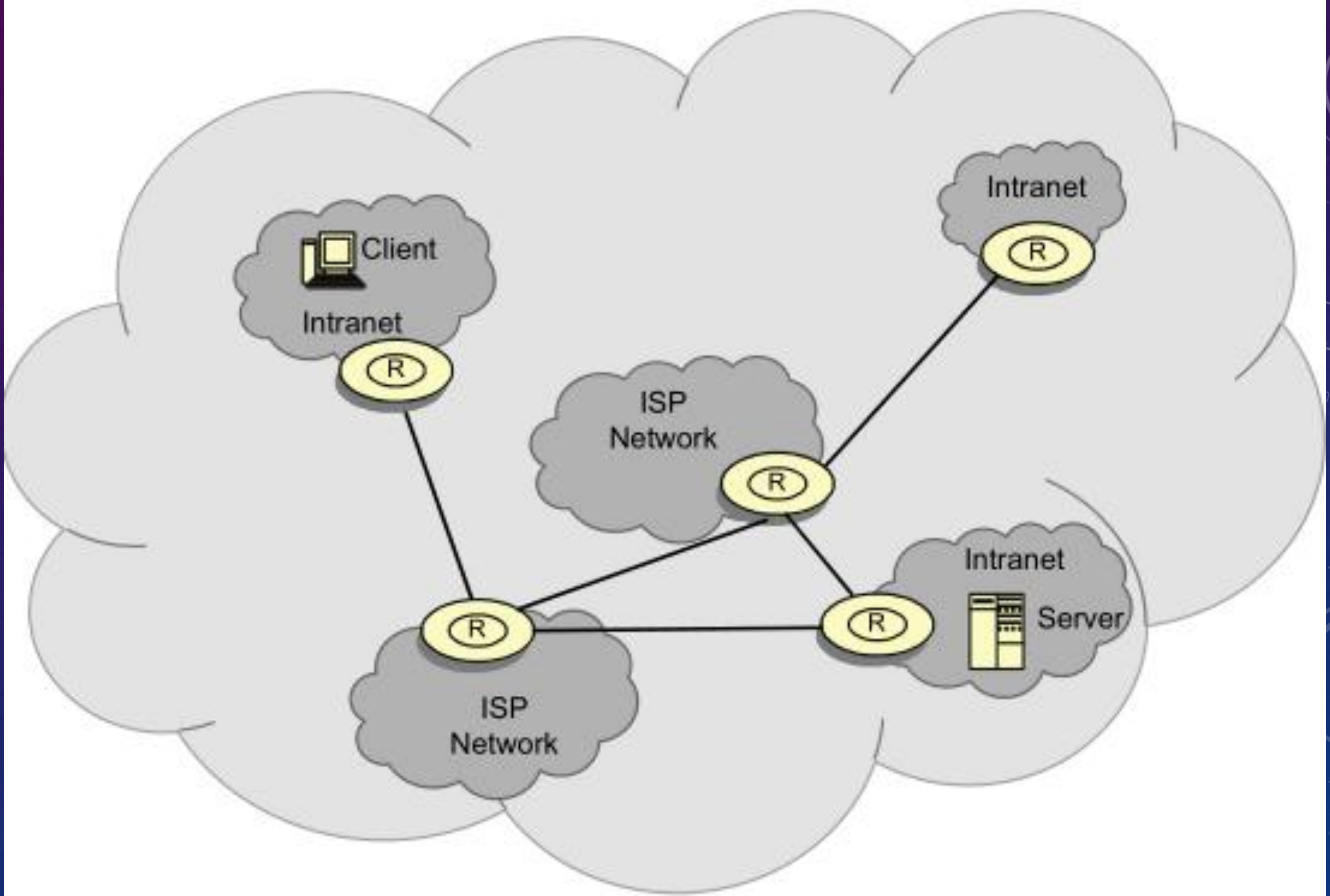



Mediator/Wrapper Architecture

# NETWORK



# NETWORK



Client
Intranet
R
Intranet
R
ISP
Network
R
Intranet
Server
R
R
ISP
Network

# TYPES OF NETWORKS

- Scale
  - LAN,WAN, MAN
- Topology
  - star, ring, complete(mesh)
- Communication Schemes
  - point-to-point, broadcaset (multi-point)
  - Microwave微波, cellular蜂窝, wireless broadband

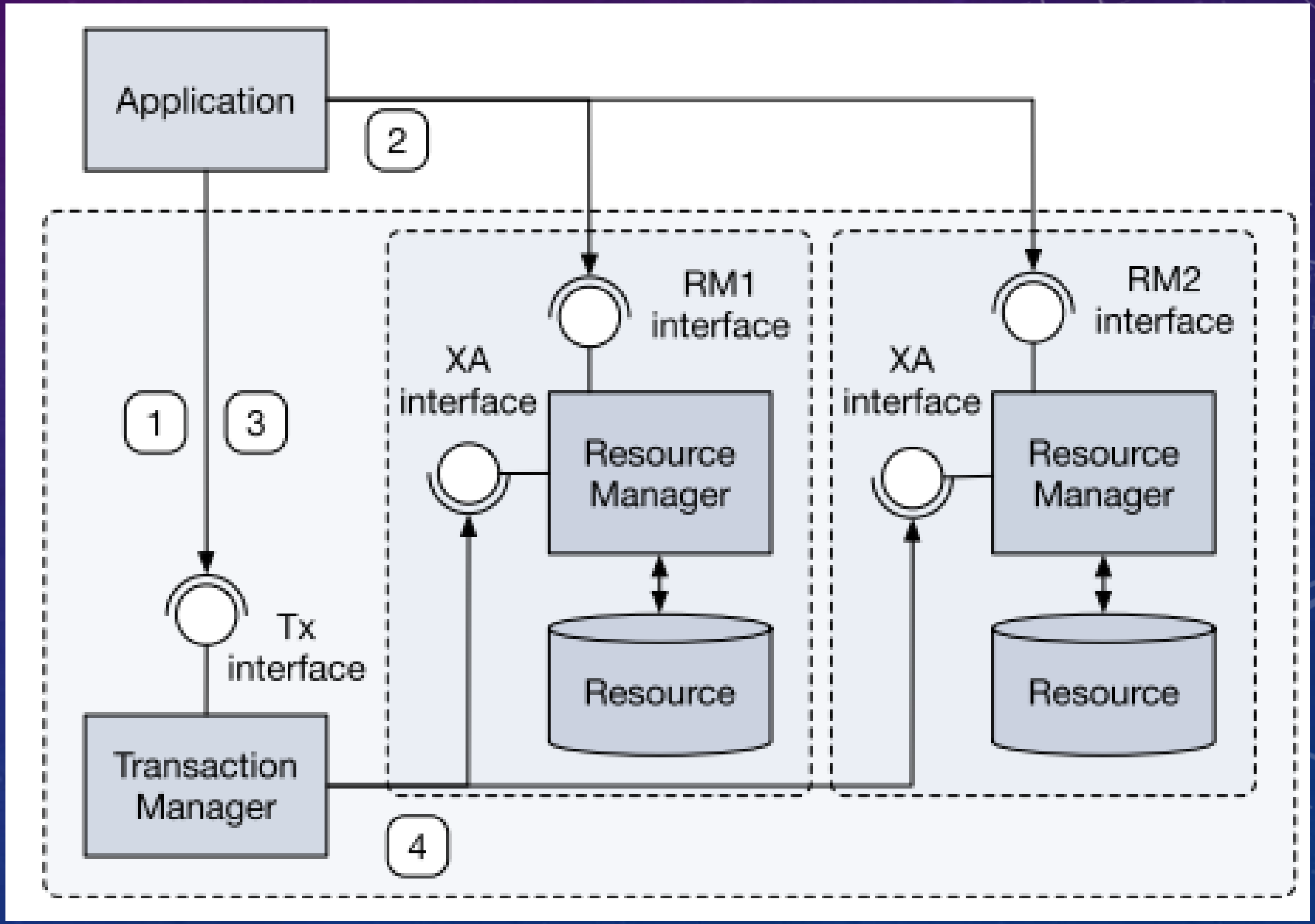
# THEOREMS OF DISTRIBUTED DB

- ACID theorem
- CAP theorem
- BASE theorem

# ACID THEOREM

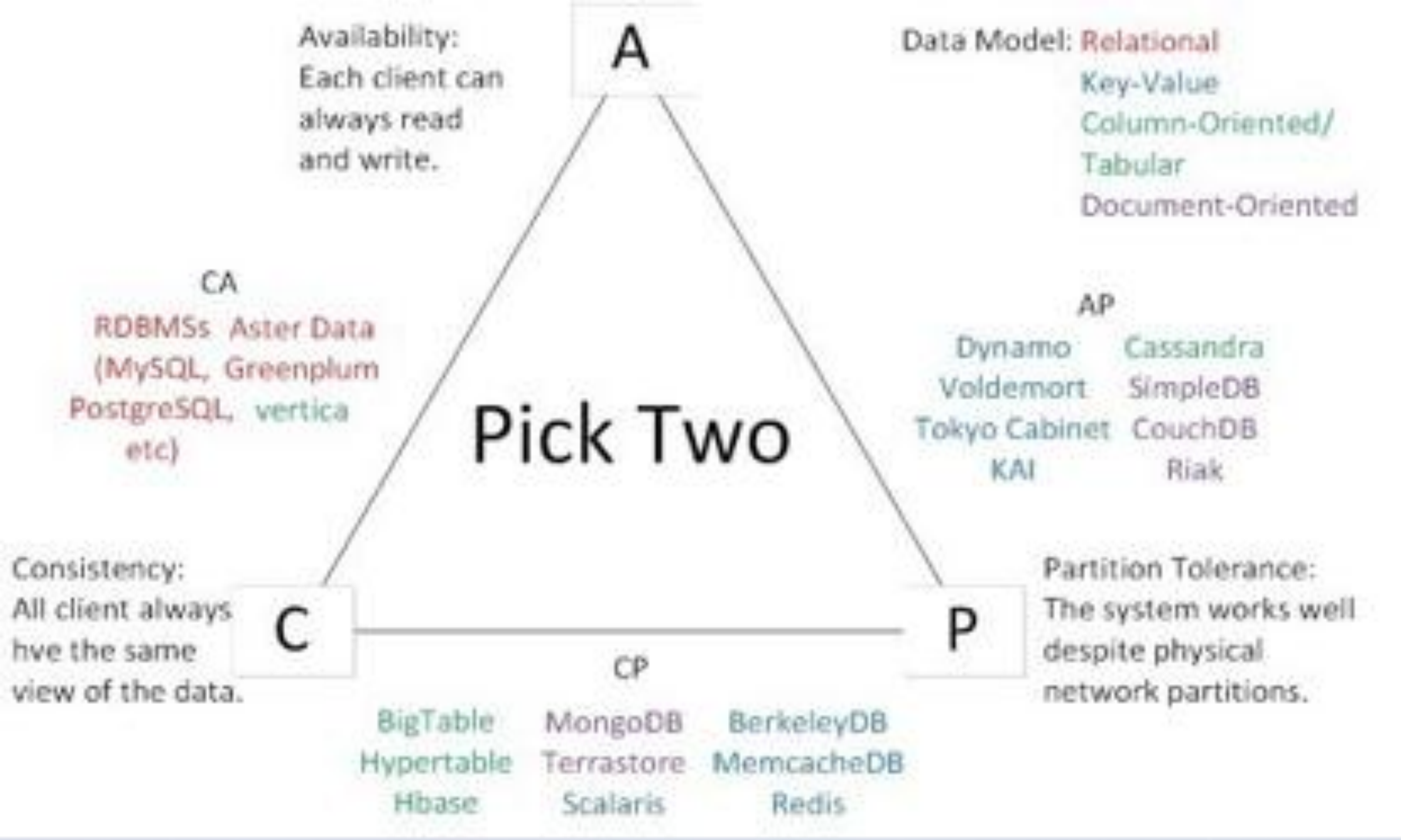
- Atomicity
- Consistency
- Isolation
- Durability

Transaction Processing System



# CAP THEOREM

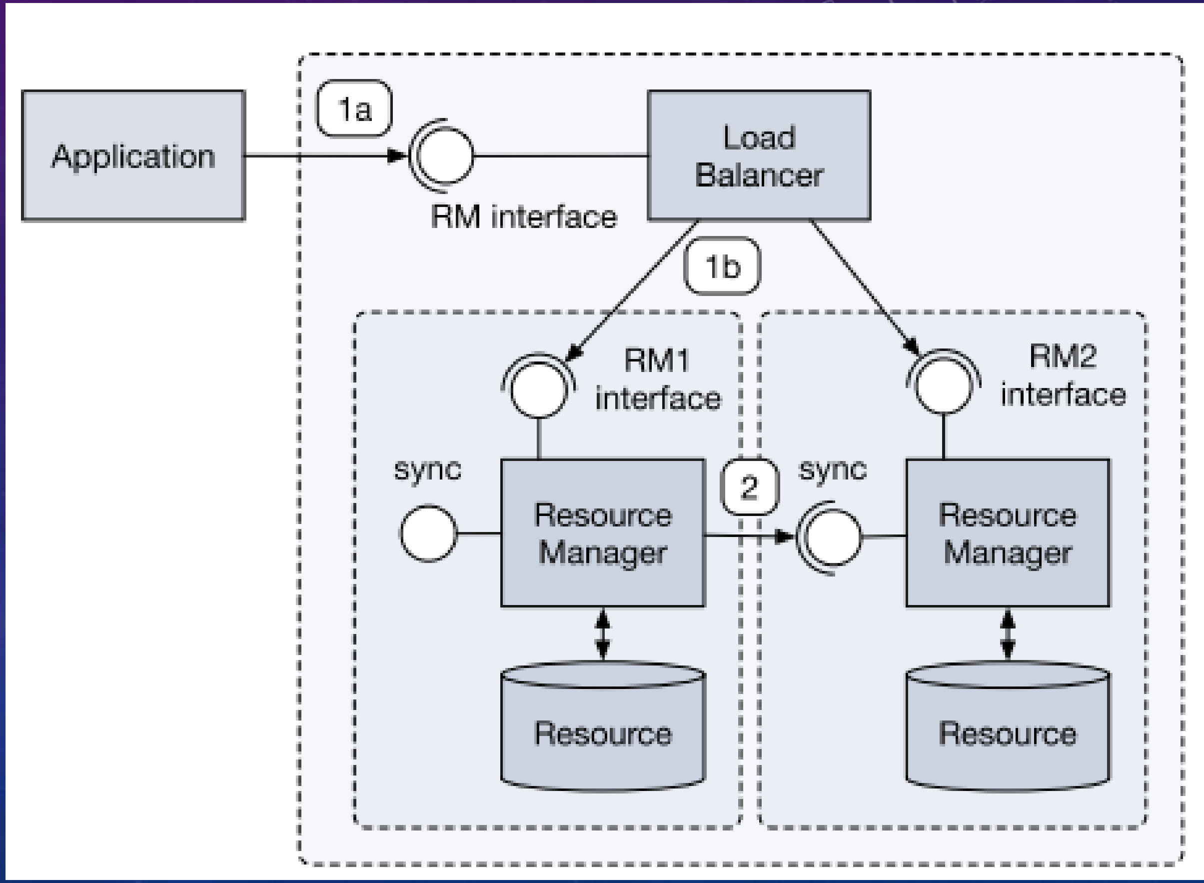
it is **impossible** for a distributed data store to simultaneously provide more than two out of the following three guarantees



# BASE THEOREM

- Basically Available
- Soft state
- Eventually consistent

## App Based BASE MODEL



# Data as a Service
# Data as a Platform

| Unit of Storage | Desc |
| --- | --- |
| Byte 字节 | 1 Byte = 8 bit |
| KB (Kilobyte, 千字节 ) | 1KB=1024 Byte |
| MB (Megabyte, 兆字节 ) | 1MB=1024KB |
| GB (Gigabyte, 吉字节 ) | 1GB=1024MB |
| TB (Trillionbyte, 太字节 ) | 1TB=1024GB |
| PB (Petabyte, 拍字节 ) | 1PB=1024TB |
| EB (Exabyte, 艾字节 ) | 1EB=1024PB |
| ZB (Zettabyte, 泽字节 ) | 1ZB=1024EB |
| YB(Yottabyte, 尧字节 ) | 1YB=1024ZB |

# DATA MINING


- Data Mining
  - Refers loosely to the process of semi-automatically analyzing large DB to find useful patterns
    - Case: beer and nappy
  - Knowledge discovery in artificial intelligence: machine learning

# DATA WAREHOUSE

- Data warehouse
  - 数据仓库
  - Gather data from multiple sources under a unified schema, at a single site
  - Then, they provide the user a single uniform interface to data

# INFORMATION RETRIEVAL

- Information Retrieval信息检索
  - Querying of unstructured textual data
  - Textual data is unstructured, unlike the rigidly structured data in relational databases
  - Textual data has grown explosively

# INFORMATION RETRIEVAL: TODAY

# Next …

# RANK OF DATABASES

354 systems in ranking, March 2020

| Rank Mar 2020 | Rank Feb 2020 | Rank Mar 2019 | DBMS | Database Model | Score Mar 2020 | Score Feb 2020 | Score Mar 2019 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1340.64 | -4.11 | +61.50 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1259.73 | -7.92 | +61.48 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 1097.86 | +4.11 | +50.01 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 513.92 | +6.98 | +44.11 |
| 5. | 5. | 5. | MongoDB | Document, Multi-model | 437.61 | +4.28 | +36.27 |
| 6. | 6. | 6. | IBM Db2 | Relational, Multi-model | 162.56 | -2.99 | -14.64 |
| 7. | 7. | ↑ 9. | Elasticsearch | Search engine, Multi-model | 149.17 | -2.98 | +6.38 |
| 8. | 8. | 8. | Redis | Key-value, Multi-model | 147.58 | -3.84 | +1.46 |
| 9. | 9. | ↓ 7. | Microsoft Access | Relational | 125.14 | -2.92 | -21.07 |
| 10. | 10. | 10. | SQLite | Relational | 121.95 | -1.41 | -2.92 |

- Relational DBMS
- Key-value stores
- Document stores
- Graph DBMS
- Time Series DBMS
- Object oriented DBMS
- RDF stores
- Search engines
- Wide column stores
- Multivalue DBMS
- Native XML DBMS
- Event Stores
- Content stores
- Navigational DBMS

# RESEARCH TOPICS

| | |
|---|---|
| 数据库实现新技术 | 云计算环境中的数据管理 |
| Web数据管理 | 查询处理与查询优化 |
| 数据流管理 | XML和半结构化数据 |
| 数据仓库和OLAP | 近似和非确定性数据库 |
| 内容与知识管理 | 数据挖掘和知识发现 |
| 元数据管理 | 数据集成和迁移 |
| 嵌入式数据库与移动数据库 | 并行和分布式数据库系统 |
| 特定领域的数据库系统 | 数据库自管理 |
| 智能用户接口技术 | 空间和时态数据库系统 |
| 多媒体数据库技术 | 数据隐私与安全 |
| 信息检索与数据库 | 协同工作技术 |
| 物联网数据管理 | |

# OBJECTIVES

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

# HISTORY OF DATABASE SYSTEMS

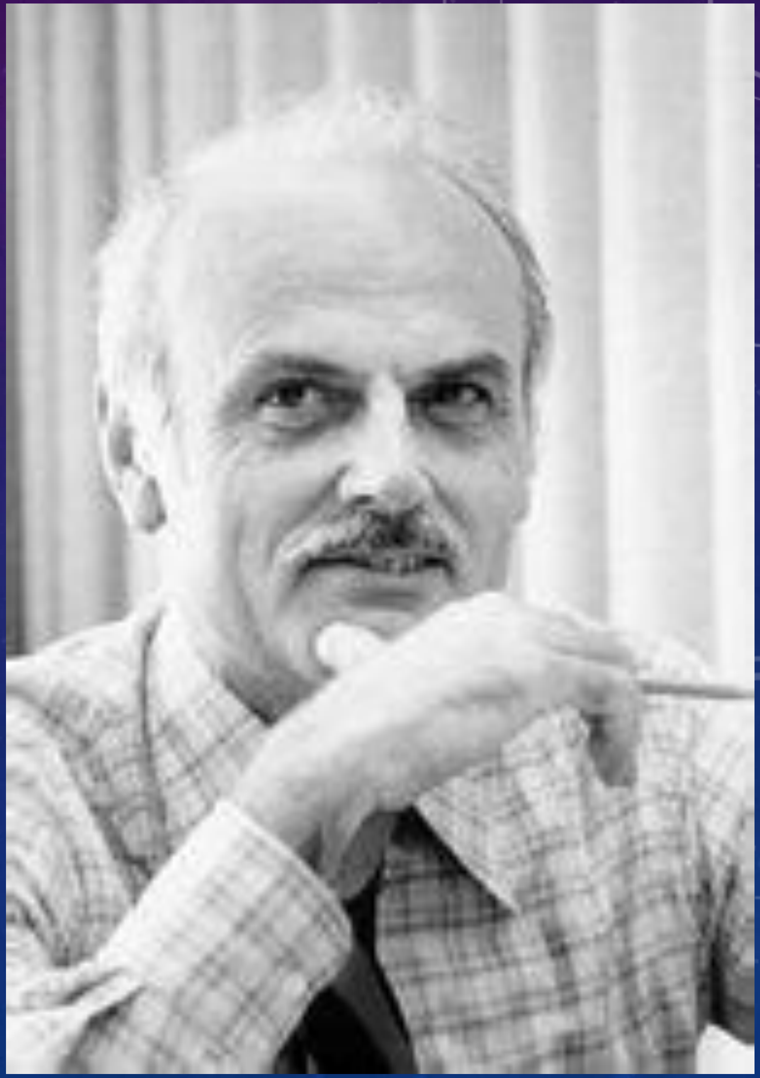
- 1950s and early 1960s: tape
- Late 1960s and 1970s
  - Hard disk, file, DB(hierarchical, network)
  - network db
    - CODASYL ,Integrated Data Store (IDS)
- **Codd**, E. F.. "A relational model of data for large shared data banks." Communications of The ACM 13.6 (1970): 377-387.

# HISTORY OF DATABASE SYSTEMS

- 1980s
  - System R: IBM
    - Astrahan, Morton M., et al. "System R: relational approach to database management." ACM Transactions on Database Systems 1.2 (1976): 97-137.
  - Ingres: BSD
  - IBM DB2, Oracle, DEC Rdb
- Early 1990s
  - Object-relational DB

# EDGAR F. CODD

- Edgar Frank Codd
  - Edgar Frank "Ted" Codd was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases
  - ACM Turing Award, 1981
  - Online analytical processing (OLAP)
  - 19 August 1923 – 18 April 2003



# HISTORY OF DATABASE SYSTEMS

- 1990s
  - WWW: DB had to support Web interface to data
- 2000s
  - XML, Xquery
  - PostgreSQL, MySQL
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, …

# SUMMARY

- The Age of Database
- Distributed Data Processing
- What is a Distributed Database System?
- Data Delivery Alternatives
- Promises of DDBSs
- Complications Introduced by Distribution
- Design Issues
- Distributed DBMS Architecture
- History

Q&A?

# 赞数字文明时代之开启

李旭东　2017

文明初叶几时真，造化阴阳始幻尘。
书简成山薪火旺，零壹遁迹智能春。
有形百载多将朽，数字千年总是新。
懵懂蹒跚别旧日，一朝奋起笑前津。

# THANKS！

leexudong@nankai.edu.cn