UNIT-1

Introduction

Outlines…

- Distributed Data Processing
- Distributed Database Systems
- Promises of DDBSs
- Complicating factors
- Problem areas

Distributed Database Systems

- Computer *network technology* emphasizes distributed (non-central) control
- Database systems seem to emphasize *centralization*
- Network systems seem to emphasize *distribution*
- Databases, however, are not really about centralizing the management of data
- Database management systems really *integrate*data and supply a *common access methodology to* data

Distributed Data Processing

- It means many things to many people
    - Distributed function (single program distributed on multiple processors)
    - Distributed computing (autonomous functions distributed on a network)
    - Networks (independent of function)
    - Multiprocessors (multiple CPUs in the same computer) etc.
- In any computer, there is always some aspect of distributed processing (e.g., CPU and I/O functions)
- We need a better definition of distributed computing to better understand distributed database computing

Distributed Data Processing

- *Distributed Computing System* is a set of autonomous processors that are interconnected by a computer network and that cooperate in performing an assigned task. Processors must be able to execute a computer program.
- Many things can be distributed
  - Processing logic
  - Function
  - Data
  - Control
- In our context, all of these are distributed

Distributed Database System

- A collection of multiple, logically interrelated databases distributed over a computer network

- A *Distributed Database Management System (DDBMS)*is the software systems that manages distributed databases and makes the distribution transparent to the user

- This course covers the theoretical foundation of implementing DDBMS's!

A Distributed Database System is NOT…

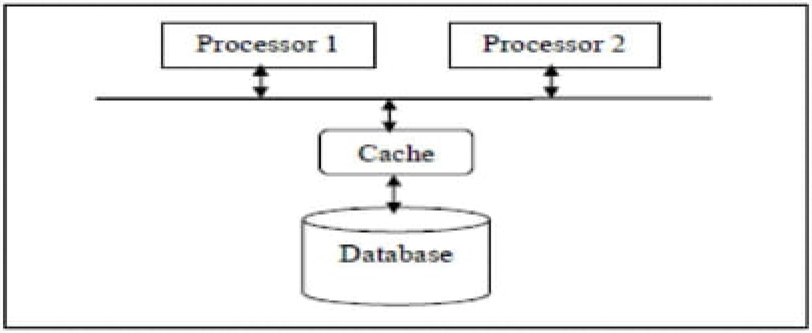
- Files would need to be logically related
- Files would need more structure for access and integration
- XML files are in the fuzzy zone
- They are often stored in files
- However, they are semi-structured
- As XML query languages and data management technologies evolve, XML files will become more integral to distributed database systems

A DDBMS is not a multiprocessing system

- Multiple computers share memory and disk
- Processors run computer programs
- Need to access data periodically
  - When data is in cache
    - Fast access
    - Expensive
    - Limited in storage -blocks get removed via LRU, etc.
  - Disks
    - Slow access
    - Cheap
    - Large amount of storage

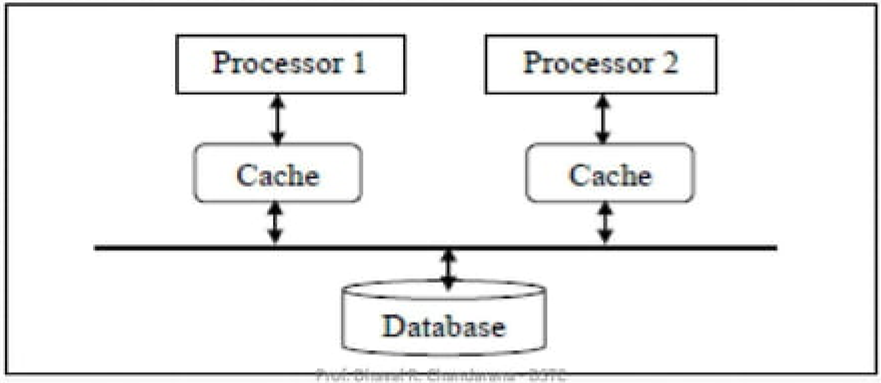
Shared Memory (Cache)

- Primary and secondary memory
- Efficient if processors are truly using same data
- Bottleneck if processors generally access their own unique data (much swapping)



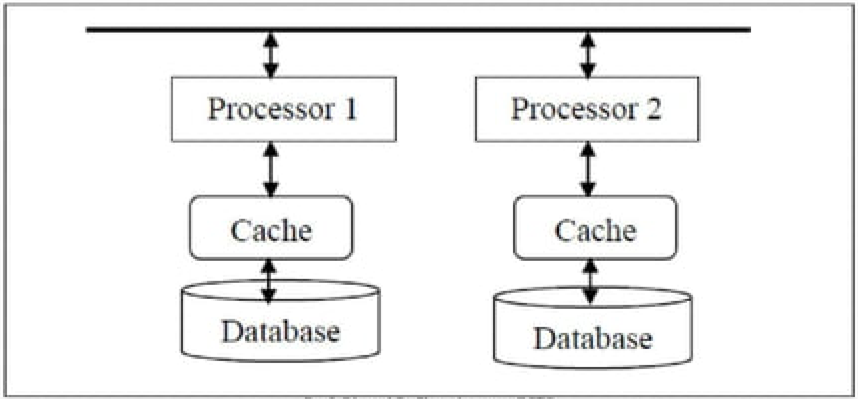
Shared Disk (Secondary Memory)

- Efficient if processors tend to used same memory, but at different times
- Inefficient if processors tend to use the same data
- Bottleneck if processors tend to go to disk often

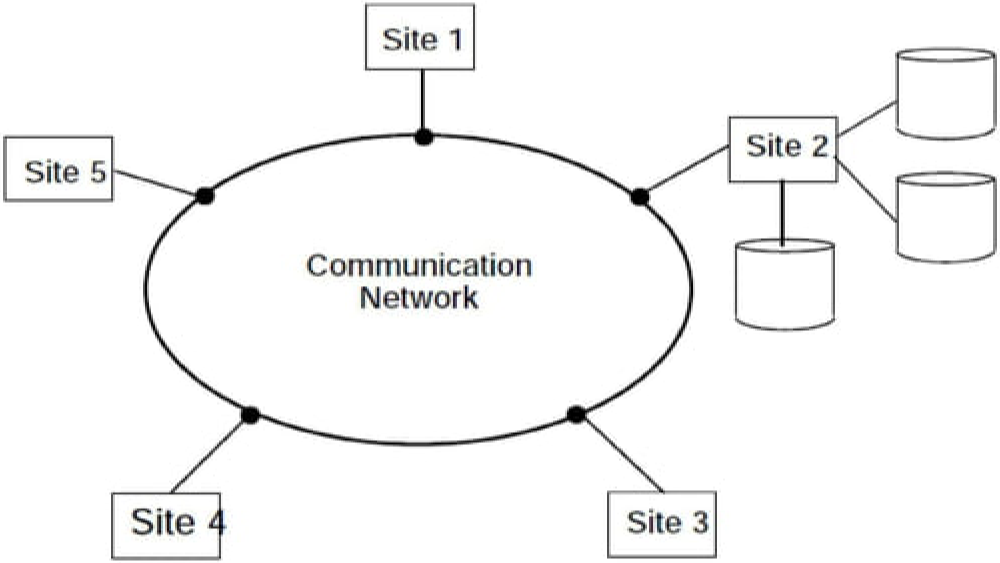


Shared Nothing

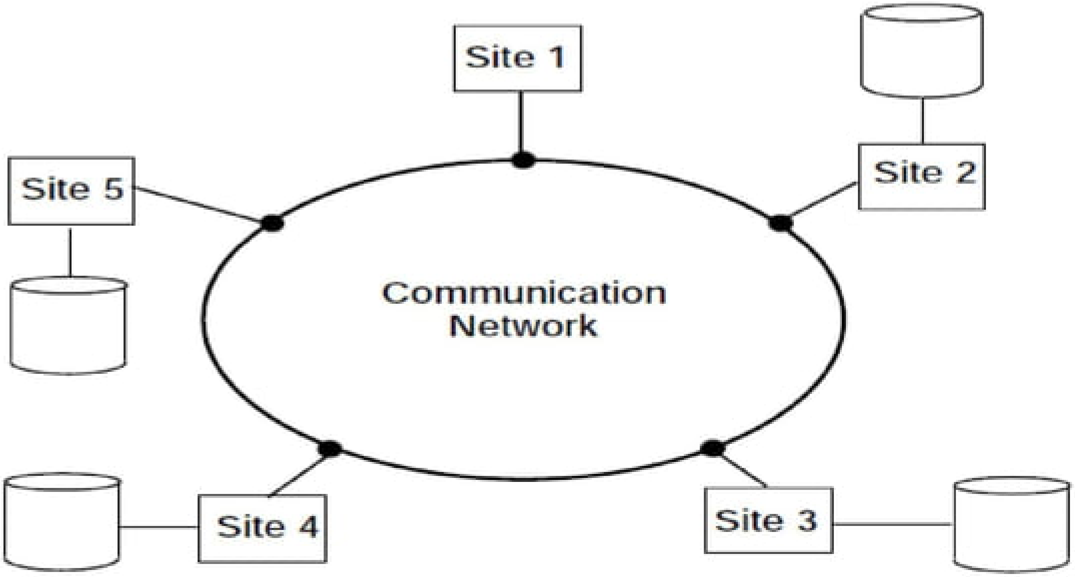
- Efficient if processors tend to retrieve data from their own databases
- Efficient if processors go to disk often
- Inefficient if processors tend to use common data and/or distributed data often



Central Database on a Network



DDBS Environment



Advantages of Distributed Database Systems

- Transparent Management of Distributed and Replicated Data
  - Transparency hides many of the lower-level implementation issues
  - Users and applications do not have to understand and manage the distribution
  - The DDBMS appears as a single DBMS
  - A single query to the DDBMS database is translated to potentially many queries on multiple DBMSs correctly
  - The effects of a single query on multiple databases are managed consistently and automatically
    - The queries are semantically correct
    - The queries are executed in the right order
  - Data replication is handled properly and automatically

Advantages of Distributed Database Systems

- Reliability Through Distributed Transactions
  - Maintains database consistency across multiple transactions
  - Multiple applications and users may execute sets of all-or-nothing queries
  - Applications and users do not "step" on each other
  - Each applications appears to be have the "complete attention" of the database
  - Effects of other user transactions are not noticed
  - This would be near impossible without a DDBMS

Advantages of Distributed Database Systems

- Improved Performance
  - Algorithms are tuned for distribution
  - Database design tuned for distribution and usage patterns
  - Based on internal DDBMS statistics, efficient query plans are calculated
  - Efficient query algorithms and optimal database design are beyond the capability of most users
  - Even if the users have these capabilities, they do not have access to internal DDBMS statistics to make effective design and query choices

Advantages of Distributed Database Systems

- Easier System Expansion
  - The next database node fits into a pre-existing architecture
  - Much of the database integration software is already in place
  - The new database node is managed consistently within the context of the other database nodes
  - Without a DDBMS, all applications needing data at the new node would need to be modified and tuned to the specifics of that database
    - Access patterns
    - Query semantics
    - Query integrity
    - Transaction management

Promises of DDBSs

1. Transparent Management of Distributed and Replicated Data

2. Reliability Through Distributed Transactions

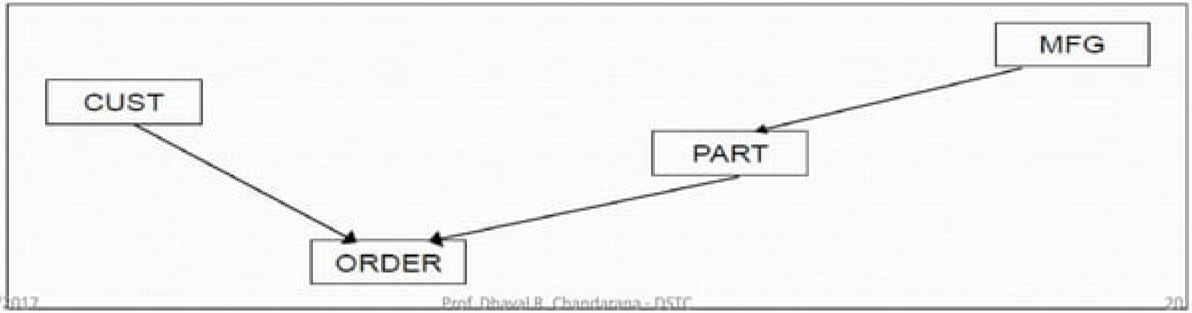
3. Improved Performance

Transparency Issues

- Distribution of data
- Replication of data
- Data independence
- Network
- Fragmentation

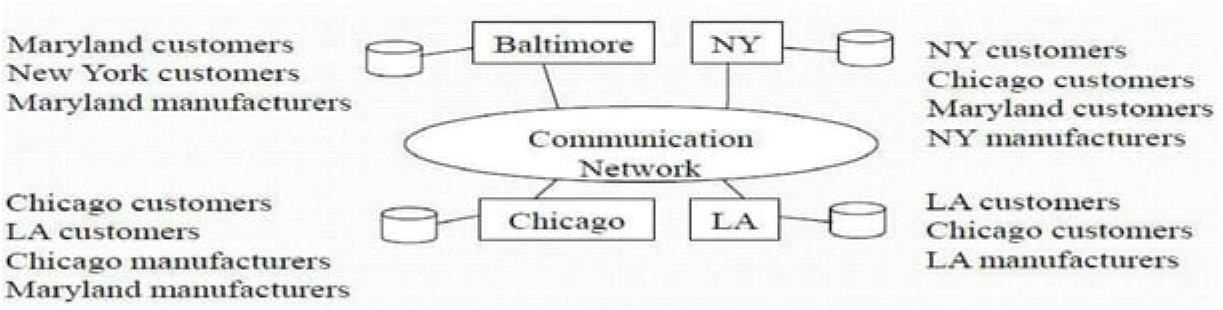
Transparent Management of Distributed and Replicated Data

- Assume the following database tables:
  - CUST(cust_no, cust_name, cust_address)
  - PART(part_no, part_name, manufacturer, cost)
  - ORDER(cust_no, part_no, quantity)
  - MFG(manufacturer, state)



Possible Query and Distribution of Data

```
SELECT cust_name, part_name, quantity
FROM CUST, PART, ORDER, MFG
WHERE CUST.cust_no = ORDER.cust_no AND
        ORDER.part_no = PART.part_no AND
        PART.manufacturer = MFG.manufacturer AND
        MFG.state = 'MD'
```



Data Independence

- Immunity of user applications to changes in the data
- Logical independence
  - Immunity to changes in the logical schema (i.e. new columns)
  - Immunity to changes of distribution!
- Physical independence
  - Immunity to changes in the physical schema
  - Details of storage structures are hidden
  - Independence from indexing schemes (e.g., random, indexed, etc.)
  - Independence from storage media (e.g., disk, tape, etc.)
  - These may change often to improve performance

Network Transparency

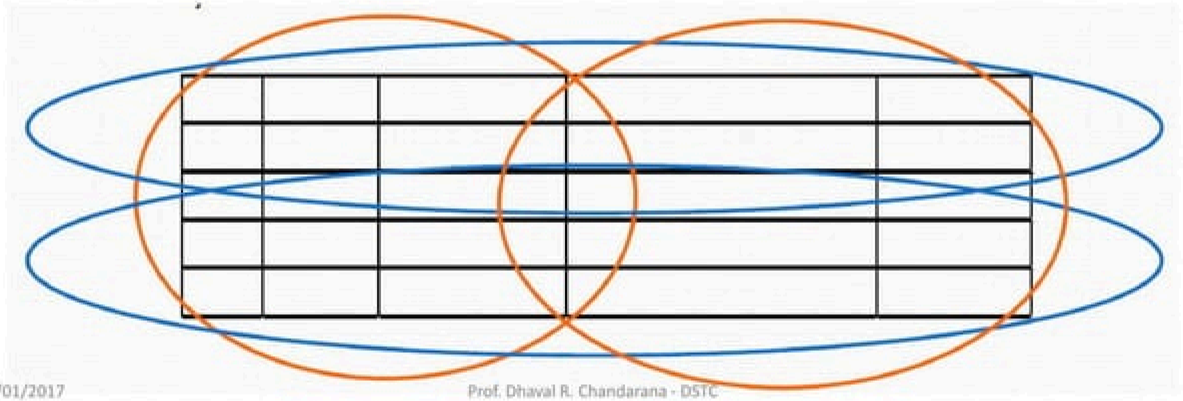
- Network needs to be shielded from the users
  - Important aspects of bandwidth and network loading are difficult for users to address on their own
  - Network topology can have a significant impact on query strategy
- Services perspective
  - No need to know where services are located
  - Services accessed in a uniform way
- DBMS perspective
  - No need to know where data is located
  - Naming transparency (independent of location)

Replication Transparency

- Data replicated for
  - Locality of reference –this will be a recurring theme
  - Reliability
  - Backup
- Should replication be transparent?
  - From user perspective, of course
  - From system perspective, transaction management would be simpler if users take control
  - However, there is a loss of flexibility if users take control because data independence is reduced
    - Users control number of copies of data
    - Users control where replicated data resides

Fragmentation Transparency

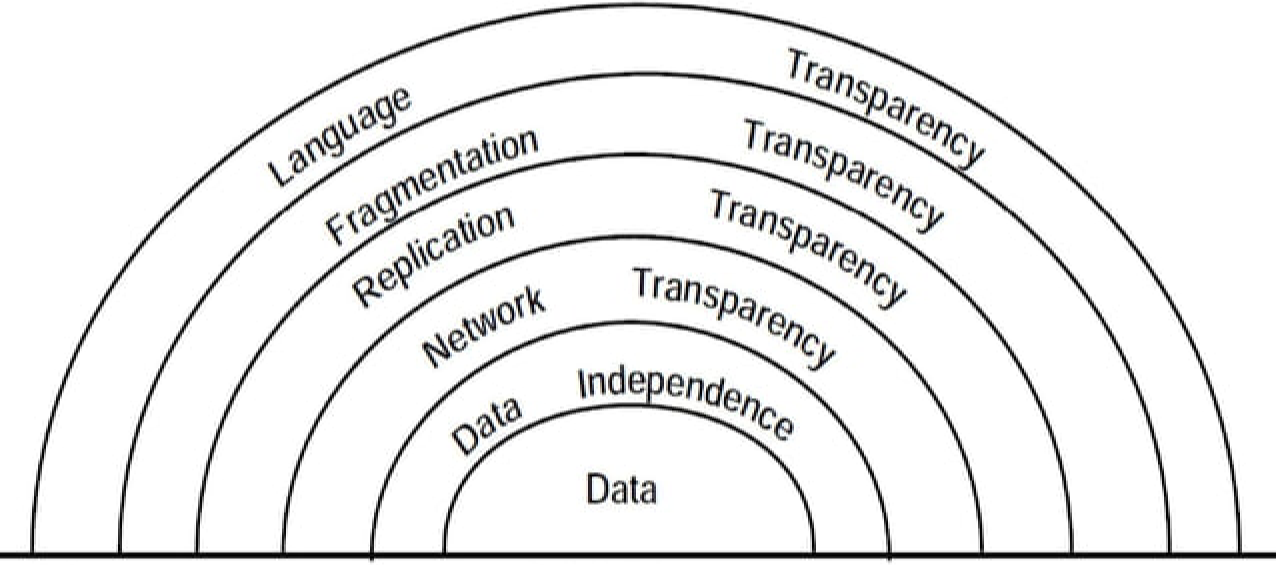
- Database tables themselves are fragmented and distributed across multiple databases for query efficiency
- Three types of fragmentation
  - Horizontal
  - Vertical
  - Hybrid of horizontal and vertical



Who Should Provide Transparency?

- Trade-off: ease of use vs. complexity
- User language
  - Responsibility is left to the compiler exclusively
  - No transparent features are implemented at lower levels
- Operating System
  - Device drivers provide some layers of transparency as system services
- DBMS
  - Provides a layer between the user and OS
  - However, in distributed environment, this is not sufficient
  - Most of this course will deal with making DDBMS systems transparent through OS features

Layers of Transparency



Reliability

- Distributed transactions
- Consistency
- Performance improvements
- Complications

Reliability Through Distributed Transactions

- Transactions ensure that groups of atomic database operations are all completed or all rejected
- ACID properties
  - Atomicity –all or nothing
  - Consistency –effects of a transaction must be repeatable (consistent)
  - Isolation –transaction operates as if it is the only process interacting with the databases
  - Durability –effects of transaction last after system crashes
- Replicated components
- Single point of failure does not make entire system unavailable
- Some data may not be available
- Users should have access to available data

Transactions Ensure Consistency

- Database is consistent at beginning and end of operation
- This includes concurrent operations (concurrency transparency)
  - This is when multiple users are updating the same data values
  - Transaction manager ensures that applications behave as if they have exclusive access to the DDBMS (isolation)
- Also, when failure occurs in the middle (failure atomicity)
  - Ensures that either all or nothing is performed
  - Example: Savings to checking account transfer

Transaction Consistency

- Requires implementation of *distributed concurrency control* and *distributed reliability protocols*
- Commercial systems provide varying degrees of support
  - Oracle supports distributed transactions
  - Sybase provides primitives for distributed transactions for users

Improved Performance

- Locality of reference
  - Local CPUs and I/O services are used
  - Remote access network traffic delays are reduced
  - Reduced contention and reduced communication overhead can be achieved through fragmentation
  - Distributed databases are arranged to get maximum benefit from distribution
- Inherent parallelism
  - Multiple queries are executed in parallel on different machines
  - Parts of the same query can be executed in parallel on different machines
- We will quantify these effects later in the course

Complicating Factors

- The problems encountered in database systems take on additional complexity in a distributed environment, even though the basic underlying principles are the same. Furthermore, this additional complexity gives rise to new problems influenced mainly by three factors.

1. First, 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. It is not essential that every site on the network contain the database; it is only essential that there be more than one site where the database resides. The possible duplication of data items is mainly due to reliability and efficiency considerations. Consequently, the distributed database system is responsible for (1) choosing one of the stored copies of the requested data for access in case of retrievals, and (2) making sure that the effect of an update is reflected on each and every copy of that data item.

Complicating Factors

2. Second, 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.
3. The third point is that 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.

Complicating Factors

- Complexity
- Cost
- Distribution of Control
- Security

Problem Areas

1. Distributed Database Design
2. Distributed Query Processing
3. Distributed Directory Management
4. Distributed Concurrency Control
5. Distributed Deadlock Management
6. Reliability of Distributed DBMS
7. Operating System Support
8. Heterogeneous Databases
9. Relationship among Problems

Distributed Database Design

- Design effects the distribution of the database
- Non-replicated vs. replicated (full or partial) partitions
- Two issues:
  - Fragmentation
  - Distribution for optimum performance
- NP-hard problem
  - Definition: The *complexity class* of *decision problems* that are intrinsically harder than those that can be solved by a *Turing machine* in *polynomial time*.
  - The only tractable approach is to apply *heuristics* that operate in polynomial time

Distributed Query Processing

- Factors
  - Distribution of data
  - Communication costs
  - Locality of data
- Take advantage of parallelism
- NP-hard problem to optimize
  - Heuristic approaches make query processing tractable

Distribution of Directory Management

- There are three levels of directories
    - Conceptual
    - Logical
    - Physical
- Directories are consulted for most database operations
- There are many issues that concern whether to distribute or centralize the directories

Distributed Concurrency Control

- Synchronization of access to distributed databases
- Maintain integrity of the system
  - Single distributed database
  - Multiple copies of the database
- Approaches
  - Pessimistic Concurrency Control
  - Optimistic Concurrency Control
- Example approaches
  - Locking
  - Timestamps

Distributed Deadlock Management

- Similar to Operating Systems deadlock management
- Well known solutions
    - Prevention
    - Avoidance
    - Detection/recovery

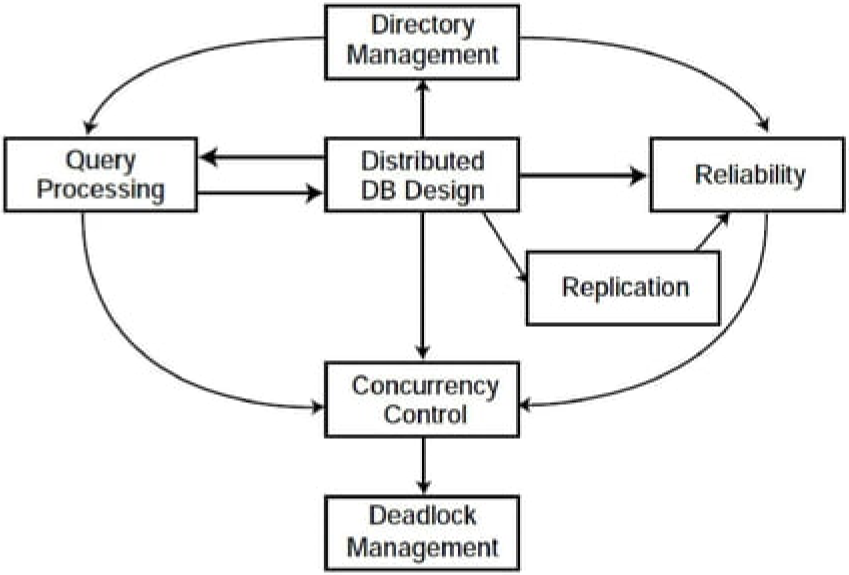
Reliability

- Failure recovery among multiple sites
- Make sure other systems are reliable and consistent
- We will explore the ARIES algorithm
- We will explore the Two Phase Commit (2PC) algorithm

Heterogeneous databases

- Sometimes called multi-databases
- Distributed databases are fully autonomous
- Usually databases already exist and the distributed database system integrates them
- Requires translations among database systems with canonical description of overall environment
- Complimentary to distributed database systems

Relationship among Problems


Directory Management
Query Processing
Distributed DB Design
Reliability
Replication
Concurrency Control
Deadlock Management