# Distributed Databases

- *Instructor*

  *Dr. Sanam Shahla Rizvi*

  *PhD in Information and Communication from Ajou University, Korea*

# DDBS Concepts

- Introduction  to DDBS.
- Advantages and disadvantages of DDBS.
- Failures in DDBS.
- Types of DDBS.
- Distributed DBMS Architecture.
- Storing data in Distributed DBS.
- Updating distributed data.
- Distributed Transactions.
- Distributed Concurrency Control.
- Distributed Recovery.
- Two-Phase Commit (2PC) Protocol.
- Restart after a Failure.
- 2PC Revisited.

# INTRODUCTION TO DDBS

In **distributed database systems** , data is physically stored across several sites and each site is typically managed by a DBMS that is capable of running independently of the other sites.

Two desirable properties for DDBS include:

- **Distributed data independence.**

  Users should able to ask queries without specifying where the referenced relations or copies or fragments of relations are located.

- **Distributed transaction atomicity.**

  Users should be able to write transactions accessing multiple sites just as writing local transactions. Effects of these transactions should be atomic.

# Advantages of DDBS

- Supports communication among geographically dispersed organizations.
- Not a single point of failure.
- Improved performance.
- Easy expansion of  distributed system.
- Reduced response time.
- Lower communication costs.

# Disadvantages of DDBS

- Increased complexity.
- Incurs much cost.
- Difficult to handle security issues at distributed locations.

# Failures in DDBS

- Transaction failures.
- Site failures.
- Media failures.
- Communication failures.

# FAILURES IN DDBS ... (CONTD.)

- **Transaction failures:** When a transaction fails, it aborts. Thereby, the database must be restored to the state it was in before the transaction started. Transactions may fail for several reasons. Some failures may be due to deadlock situations or concurrency control algorithms.

- **Site failures:** Site failures are usually due to software or hardware failures. These failures result in the loss of the main memory contents. In distributed database, site failures are of two types:

  1. ***Total Failure*** where all the sites of a distributed system fail.

  2. ***Partial Failure*** where only some of the sites of a distributed system fail.

# FAILURES IN DDBS … (CONTD.)

- **Media failures:** Such failures refer to the failure of secondary storage devices. The failure itself may be due to head crashes, or controller failure. In these cases, the media failures result in the inaccessibility of part or the entire database stored on such secondary storage.

- **Communication failures:** Communication failures are failures in the communication system between two or more sites. As such, messages from one site won't reach the other sites and will therefore be lost. The reliability protocols then utilize a timeout mechanism in order to detect undelivered messages. A message is undelivered if the sender doesn't receive an acknowledgment. The failure of a communication network to deliver messages is known as **performance failure**.

# Types of DDBS

- Homogeneous Distributed Database Systems.
- Heterogeneous Distributed Database Systems.

# DDBMS Architecture

- Client Server System.
- Collaborating Server Systems.
- Middleware Systems.

# DDBMS Architecture … (Contd.) Client Server System

- A client server system has one or more client and server processes, and a client process can send a query to any one server process.
- Client Server architecture is the most famous architecture due to:
    1. Clean separation of functionality.
    2. User friendly interface at client machines.
    3. Avoid underutilization of server machines.

# DDBMS Architecture … (Contd.) Collaborating Server Systems

- Collaborating Server Systems allow a single query to span multiple servers.
- Collection of servers cooperatively executing the transaction spanning multiple servers.
- Server generates sub queries to be executed at other servers and then computes answers to original query.

# DDBMS Architecture … (Contd.) Middleware Systems

- In middleware system a special server is dedicated for managing queries and transactions spanning multiple servers. This special server is called Middleware.
- The remaining servers handle queries and transactions locally.

# Storing data in Distributed DBS

- As in distributed database systems, data is located at several sites therefore while processing a query in which a remotely located database server is involved then the cost of message passing goes exceptionally high.
- In order to reduce this message passing cost. This is achieved in two ways:

    1. Fragmentation.
    2. Replication.

- Fragmentation consists of breaking a relation into smaller relations or fragments and storing the fragments.
- In **Horizontal Fragmentation** each fragment consists of a **subset of rows** of the original relation.
- In **Vertical Fragmentation**, each fragment consists of a **subset of columns** of the original relation.

# Replication

- Replication means that several copies of a relation or relation fragment.
- An entire relation can be replicated at one or more sites. Similarly. one or more fragments of a relation can be replicated at other sites.
- Replication is preferred because of:
  - Increased Availability of data.
  - Faster Query Evaluation.
- The two types of replication are:
  - Asynchronous Replication.
  - Synchronous Replication.

# Asynchronous Replication

- Technique for replicating data between databases (or file systems) where the system being replicated does not wait for the data to have been recorded on the duplicate system before proceeding.
- Asynchronous Replication has the advantage of speed, at the increased risk of data loss during due to communication or duplicate system failure.

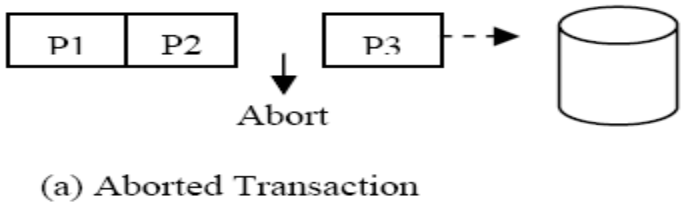
# Synchronous Replication

- Technique for replicating data between databases (or file systems) where the system being replicated does wait for the data to have been recorded on the duplicate system before proceeding.
- Synchronous Replication has the advantage that it is guaranteed that the duplicate system has a copy of the data, but the disadvantage that the primary system must wait for the secondary system before proceeding, leading to an increased response time.
- Because of the increased response time and communication delays, synchronous replication is often impractical unless the secondary system is physically located close to the primary system.

# Fundamentals of Transaction Management

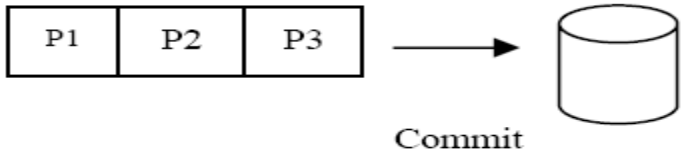
- Transaction consists of series of operations performed on the database.
- Transaction is the unit of consistency and reliability.
- Every transaction that starts must have to terminate. Two possibilities of how a transaction may terminate are:

1. **Aborts** if failure occurs.




(a) Aborted Transaction

2. **Commits** in case of successful execution.




(b) Committed Transaction

# Properties of Transaction

- **Atomicity:** Either all of the actions related to transaction are completed or none of them is carried out. The recovery of transaction is split into two types:

  1. Transaction recovery.
  2. Crash recovery.

- **Consistency:** Falls under the area of concurrency control. Consistent state of database must be maintained.

# Properties of Transaction…(Contd.)

- **Isolation:** Each transaction should see the consistent database at all the times. No other transaction can read or modify data that is being modified by another transaction. Two types of problems arise if this property is not maintained:

1. Lost Updates.

2. Cascading Aborts.

| Time | T1 | T2 |
|---|---|---|
| Time 1 | Read x | |
| Time 2 | X=x*2 | Read x |
| Time 3 | Write x | x=x+20 |
| Time 4 | | Write x |

(a) Lost Updates

| Time | T1 | T2 |
|---|---|---|
| Time 1 | (…) | (…) |
| Time 2 | (…) | (…) |
| Time 3 | ABORT | ABORT |

(b) Cascading Abort

# Properties of Transaction…(Contd.)

- **Durability:** Once transaction commits its results are permanent and cannot be erased from the database whether system crashes or aborts of other transactions.

# Distributed Transactions

- In a distributed DBMS, a given transaction is submitted at some one site but it can access data at other sites as well.
- A terminology that is extensively used in distributed transactions is sub transaction. It is defined as an activity of transaction at a given site.
- When a transaction is submitted at some site, the transaction manager at that site breaks it up into a collection of one or more sub transactions that execute at different sites, submits them to transaction managers at other sites, and coordinates their activity.
- The two core issues in distributed transactions are:
  - Distributed Concurrency Control.
  - Distributed Recovery.

# Distributed Concurrency Control

- In concurrency control it is determined that when should locks are obtained and released.
- Lock management can be distributed across sites in many ways:
  - **Centralized:** A single site is in charge of handling lock and unlock requests for all objects.
  - **Primary Copy:** One copy of object is designated the primary copy. All requests to lock or unlock a copy of this object are handled by the lock manager at the site where the primary copy is stored, regardless of where the copy itself is stored.
  - **Fully Distributed:** Requests to lock or unlock a copy of an object stored at a site are handled by the lock manager at the site where the copy is stored.

# Distributed Recovery

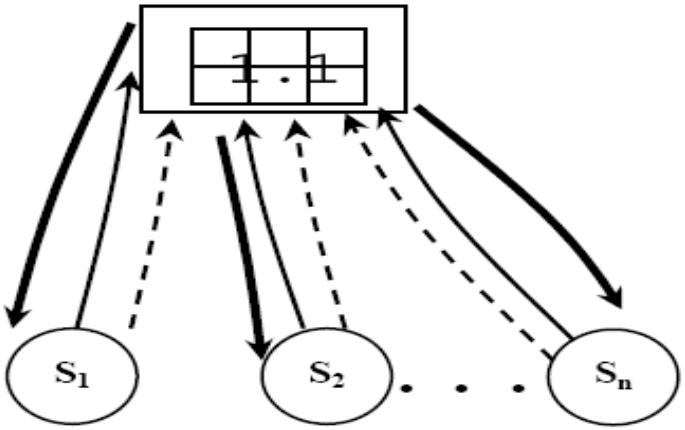
- Recovery in a distributed DBMS is much complicated:
  - Failure of communication links and failure of a remote site at which a transaction is executing.
  - Either all sub transactions of a given transaction must commit or none must commit. This property is achieved using a **Commit Protocol**. The most widely used commit protocol is **Two-Phase Commit Protocol.**

# Two-Phase Commit Protocol (2PC)

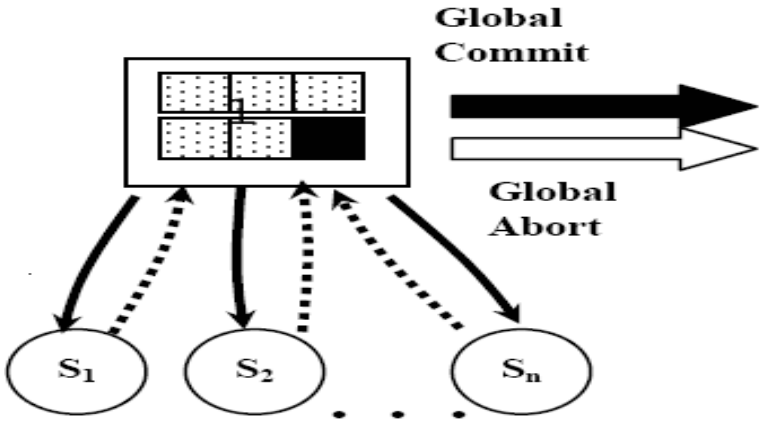
- 2PC protocol has two types of nodes to complete its processes:
    1. Coordinator.
    2. Subordinate.
- 2PC protocol involves TWO phases:
    1. PREPARE phase. (Voting Phase)
    2. A decision-making phase or Termination Phase:
        - ->COMMIT message.
        - ->ABORT message.
- Three different methods exist to carry out the 2PC:
    1. Centralized 2PC.
    2. Linear 2PC.
    3. Distributed 2PC.

# Two-Phase Commit Protocol (2PC)… (Contd.) The Centralized 2PC Protocol

- Communication via coordinator's process only. No communication among subordinates is allowed. Two steps are involved in this method are:



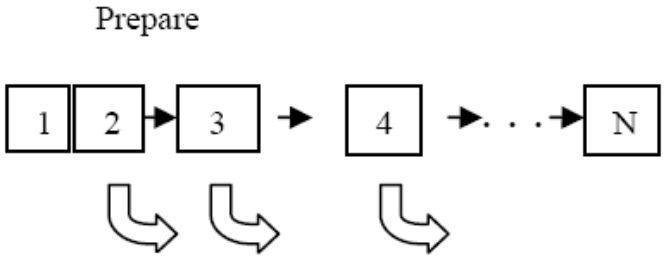

a. Step One in Centralized 2PC



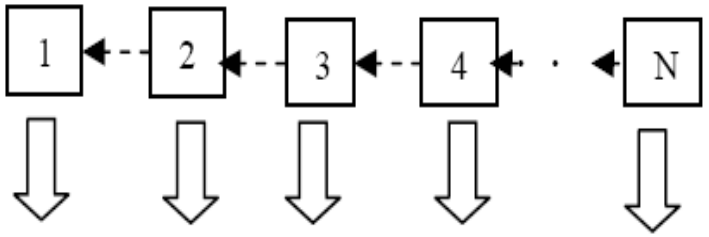

b. Step Two in Centralized

# Two-Phase Commit Protocol (2PC)… (Contd.) The Linear 2PC Protocol

- Subordinates can communicate with each other.
- Time required to complete the transaction is longer then centralized or distributed methods.
- Two phase process.



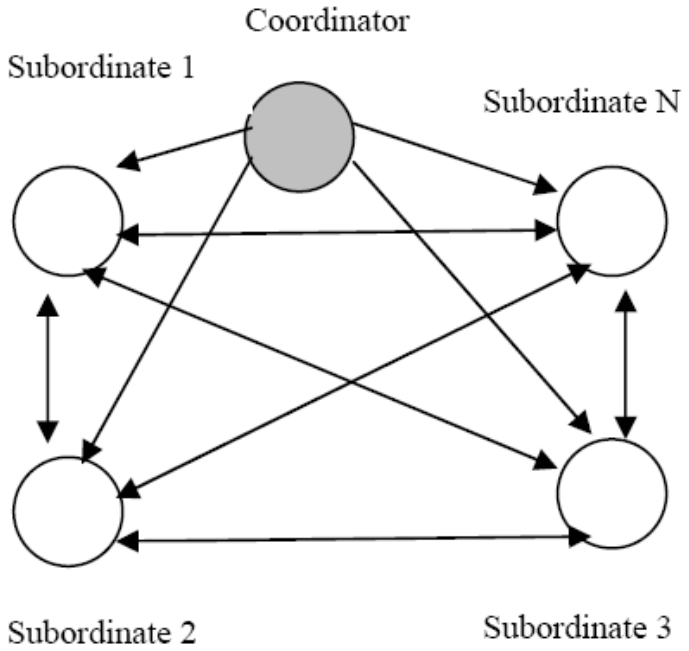
(a) Linear 2PC Phase 1



(b) Linear 2PC Phase 2

# Two-Phase Commit Protocol (2PC)… (Contd.) The Distributed 2PC Protocol

- All the participating nodes can communicate with each other.
- Each node maintains the list of all other participating nodes.
- The only purpose of coordinator is to issue a PREPARE message to all of the subordinates.



# Restart after a Failure

- When a site comes back after a crash, we invoke a recovery process that reads the log and processes all transactions executing the commit protocol at the time of the crash.
- Recovery process is carried as follows:
  - If we have a commit or abort log record for a transaction T, we redo or undo T, respectively. If this site is the coordinator site, which can be determined from the commit or abort log record, we must periodically resend as there may be other link or site failures in the system a COMMIT or ABORT to each subordinate until we receive an ack. After we receive an ack from all subordinates we write an end log record for transaction T.
  - If we have a prepare log record for T but no commit or abort log record, then this site is a subordinate and the coordinator can be determined from a prepare record.

# Restart after a Failure … (Contd.)

- If we have no prepare, commit or abort log record for transaction T, then it is certain to say that T could not have been voted to commit before the crash, so we can unilaterally abort and undo T and write an end log record. In this case, we have no way to determine whether the current site is the coordinator or a subordinate for T.

# Two- Phase Commit Revisited

- The three listed scenarios where refinement is required are:
  - The ack messages in 2PC are used to determine when a coordinator can "forget" about a transaction T. Until the coordinator knows that all subordinates are aware of the commit or abort decision for T, it must keep information about T in the transaction table.
  - If the coordinator site fails after sending out prepare messages but before writing a commit or abort log record, when it comes back, it has no information about the transaction's commit status prior to crash. However it is still free to abort the transaction unilaterally . If another site inquires about the status of transaction, the recovery process responds by sending an abort message.
  - If a sub transaction does no updates, it has no changes to either redo or undo; in other words its commit or abort status is irrelevant.

# 2PC Revisited … (Contd.)

- Refinements to first two scenarios are:
  - When a transaction aborts a transaction T, it can undo T and remove it from the transaction table immediately.
  - Similarly if a subordinate receives an abort message it need not send an ack message.
  - Because the coordinator is not waiting to hear from subordinates after deciding to abort a transaction, the names of subordinates need not to be recorded in the abort log record for the coordinator.
- Refinements to the third scenario are:
  - If a sub transaction does no updates, the subordinate can respond to a prepare message from the coordinator with a reader message. The subordinates writes no log record in this case.

# 2PC Revisited … (Contd.)

- When a coordinator receives a reader message, it treats the message as a yes vote, but with the optimization that it does not send any more messages to the subordinate, because the subordinates commit or abort status is irrelevant.
- If all sub transactions, including the sub transaction at the coordinator's site send a reader message, we do not need the second phase of the commit protocol.