**Concurrency control** ensures that multiple transactions can execute concurrently without leading to inconsistent data states. In a distributed environment, concurrency control becomes more complex due to data and transactions' distribution. Some techniques used for concurrency control include:

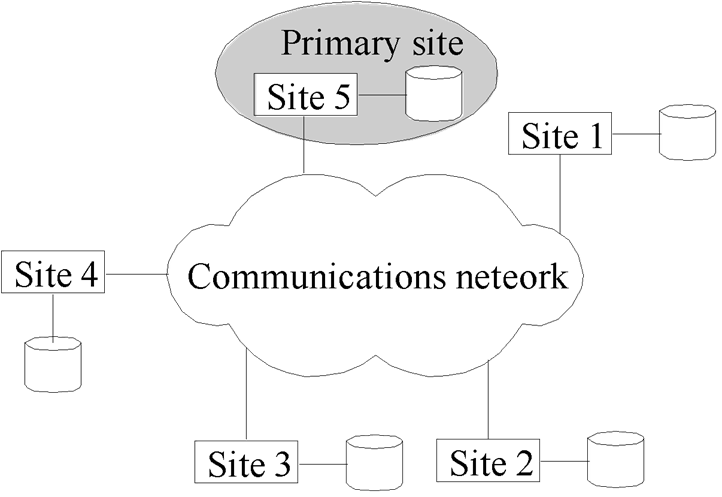
1. *Locking Mechanisms:* Acquiring locks on data items to prevent other transactions from accessing or modifying them concurrently. Locks can be of different types like read locks, write locks, etc.
2. *Timestamp Ordering:* Transactions are assigned unique timestamps, and their execution order is determined based on these timestamps. This technique ensures serializability by maintaining a total order of transactions.
3. *Optimistic Concurrency Control:* Transactions are allowed to execute without acquiring locks initially. However, before committing, the system checks if the transaction's changes conflict with other transactions. If conflicts are detected, appropriate actions are taken to resolve them.

**Recovery** mechanisms ensure that the database can recover to a consistent state after a failure or crash. Key components of recovery in DDBMS include:

1. *Logging:* Every operation performed on the database is logged in persistent storage. These logs contain info about transactions, their operations, & other relevant details.
2. *Checkpointing:* Periodic checkpoints are taken to save the current state of the database. Checkpoints reduce the time required for recovery by providing a consistent starting point.
3. *Undo and Redo Operations:* During recovery, transactions are either undone (rollback) or redone (replay) based on the logged information. Undo operations are applied to transactions that were not committed at the time of failure, while redo operations ensure that committed transactions are reapplied to bring the database to a consistent state.
4. *Distributed Recovery Protocols:* Recovery protocols need to handle the distributed nature of the system. This involves coordination among distributed sites to ensure global consistency during recovery processes.

**Dealing with Challenges in Distributed Databases:**

1. *Multiple Copies of Data Items:* Concurrency control mechanisms must ensure global consistency across multiple copies of data items. Recovery mechanisms need to recover all copies of data items and maintain consistency after recovery.
2. *Failure of Individual Sites:* Database availability should not be compromised by the failure of one or more sites. Recovery schemes must restore failed sites before they become available for use.
3. *Communication Link Failure:* Robust communication protocols and strategies for handling network partitions are necessary to maintain database availability.
4. *Distributed Commit:* Transactions may be fragmented and executed across multiple sites, requiring a distributed commit protocol like a two-phase or three-phase commit.
5. *Distributed Deadlock:* Distributed deadlock detection & resolution mechanisms are necessary to resolve them efficiently w/o impacting system performance.
6. *In-Doubt Transactions:* Recovery algorithms must determine the commit or abort status of in-doubt transactions by contacting other sites. Lock information can be logged to facilitate recovery and ensure transaction consistency.

**Primary Site Technique:**

One site is chosen as the primary site to manage transaction coordination, including concurrency control and commit.

1. *Transaction Management:*

- All transaction-related activities, like locking and releasing data items, are handled by this primary site.

- Two-phase locking is commonly used, ensuring serializability if all sites follow the two-phase policy.

1. *Advantages:*

- Simple extension of centralized two-phase locking.

- Data items are locked only at one site but can be accessed from any site.

1. *Disadvantages:*

- Risk of overloading the primary site.

- System becomes inaccessible if the primary site fails.

1. *Recovery from Coordinator Failure:*

- Without a backup site: Abort and restart active transactions at all sites, elect a new coordinator, and resume transaction processing.

- With a backup site: Suspend active transactions, promote the backup site to primary, designate a new backup, and resume processing.

- If both primary and backup sites fail: Use an election process to select a new coordinator.

**Primary Copy Technique:**

Instead of a site, a data item partition is designated as the primary copy which is locked.

*Advantages:* Distribution of primary copies reduces the load on individual sites.

*Disadvantages:* Identifying primary copies and maintaining a distributed directory is complex.

**Concurrency Control Based on Voting:**

No primary coordinator; lock requests are sent to sites holding the data item.

When a transaction needs a data item, it asks the sites that have it for permission to use it.

If most of the sites agree, the transaction can proceed with the data item. The decision is shared with all sites. To prevent waiting forever, there's a time limit. If no response comes in time, the transaction is canceled.

**Commit protocols** ensure that transactions are either committed at all sites or aborted at all sites, avoiding inconsistencies across a distributed system. The 2PC is commonly used but has drawbacks. The 3PC is more complex & costly but addresses some issues of 2PC.

**Two-Phase Commit Protocol (2PC)**

Ensures atomicity of transactions across multiple sites. All sites must either commit or abort a transaction; partial commitment or abortion is not permissible.

The coordinator triggers the protocol after the transaction reaches its final step.

Fail-Stop Model: Assumes failed sites stop functioning without causing further harm, like sending incorrect messages.

Transaction initiated at site Si, with coordinator Ci at Si.

1. *Phase 1:*

Coordinator instructs all participants to prepare for committing transaction Ti. Ci logs <prepare T>, sends those messages to all involved sites.

Upon reception, local transaction managers determine commit feasibility:

- If unable to commit, log <no T> and send abort T message to Ci.

- If ready to commit, log <ready T>, force all T records to stable storage, and notify Ci.

1. *Phase 2:*

Transaction can commit if Ci receives <ready T> msgs from all participants; else, it aborts.

Coordinator logs <commit T> or <abort T> irreversibly upon receiving all decisions and notifies participants.

*Handling Failures:-*

1. *Coordinator Failure:-*

- Active sites with <commit T> or <abort T> records act accordingly.

- Sites lacking <ready T> reject subsequent <prepare T> from Ci.

- Active sites await Ci recovery if <ready T> is present but no decision.

1. *Network Partition:-*

- No impact if coordinator and participants remain in the same partition.

- Otherwise, non-coordinator partition proceeds with failure protocols.

- Coordinator partition operates normally assuming other partition failure.

Notes:

1. *Practicality:* Widely used due to simplicity despite some drawbacks.
2. *Atomicity Assurance:* Ensures transactions are either fully committed or fully aborted.
3. *Failures Handling:* Addresses both coordinator and network partition failures.
4. *Blocking Issue:* Active sites may experience delays awaiting coordinator recovery.
5. Simplicity & widespread usage make 2PC preferred despite 3PC's advantages.

**Three Phase Commit (3PC):**

*Assumptions:-*

1. No network partitioning (network remains fully connected, all sites can communicate with each other without any partitions or disconnections).
2. There must be atleast one operational site in system to ensure protocol can proceed.
3. System can tolerate failure of up to a max of K sites be it participants or coordinator.

*Phases:-*

1. All sites are ready to commit if instructed.
2. The coordinator makes a pre-commit decision, records it at many sites, & informs participants who take action based on the decision.
3. The coordinator collects acknowledgments, commits if pre-committed, and informs participants to commit.

*Handling Site Failure:-*

Upon recovery, a site checks its log:

1. If it contains a commit or abort record, no action is needed.
2. If it contains a pre-commit record, site consults coordinator for transaction's fate:
   1. If aborted, the site undoes the transaction.
   2. If committed, it redoes the transaction.
   3. If still in pre-commit state, it resumes the protocol.
3. If no record of readiness for a transaction exists, the site aborts the transaction.

*Advantages:-*

1. Knowledge of pre-commit decisions allows committing despite coordinator failure.
2. Avoids blocking problems as long as fewer than K sites fail.

*Drawbacks:-*

1. Higher overheads compared to 2PC.
2. Assumptions may not always hold true in practical scenarios.