**Assignment No – 2**

**Implementation of Two Phase Commit Protocol in Distributed Databases**

**Problem Statement:**

Implementing Two Phase Commit protocol in distributed database.

**Theory:**

**Introduction:**

In [transaction processing](https://en.wikipedia.org/wiki/Transaction_processing), [databases](https://en.wikipedia.org/wiki/Database), and [computer networking](https://en.wikipedia.org/wiki/Computer_networking), the two-phase commit protocol (2PC) is a type of [atomic commitment protocol](https://en.wikipedia.org/wiki/Atomic_commit) (ACP). It is a [distributed algorithm](https://en.wikipedia.org/wiki/Distributed_algorithm) that coordinates all the processes that participate in a [distributed atomic transaction](https://en.wikipedia.org/wiki/Distributed_transaction) on whether to [commit](https://en.wikipedia.org/wiki/Commit_(data_management)) or abort (roll back) the transaction (it is a specialized type of [consensus](https://en.wikipedia.org/wiki/Consensus_(computer_science)) protocol). The protocol achieves its goal even in many cases of temporary system failure (involving either process, network node, communication, etc. failures), and is thus widely utilized. However, it is not resilient to all possible failure configurations, and in rare cases, user (e.g., a system's administrator) intervention is needed to remedy an outcome. To accommodate recovery from failure (automatic in most cases) the protocol's participants use [logging](https://en.wikipedia.org/wiki/Server_log) of the protocol's states. Log records, which are typically slow to generate but survive failures, are used by the protocol's [recovery procedures](https://en.wikipedia.org/wiki/Recovery_procedure). Many protocol variants exist that primarily differ in logging strategies and recovery mechanisms. Though usually intended to be used infrequently, recovery procedures compose a substantial portion of the protocol, due to many possible failure scenarios to be considered and supported by the protocol.

In a "normal execution" of any single [distributed transaction](https://en.wikipedia.org/wiki/Distributed_transaction), i.e., when no failure occurs, which is typically the most frequent situation, the protocol consists of two phases:

1. The commit-request phase (or voting phase), in which a coordinator process attempts to prepare all the transaction's participating processes (named participants, cohorts, or workers) to take the necessary steps for either committing or aborting the transaction and to vote, either "Yes": commit (if the transaction participant's local portion execution has ended properly), or "No": abort (if a problem has been detected with the local portion), and
2. The commit phase, in which, based on voting of the cohorts, the coordinator decides whether to commit (only if all have voted "Yes") or abort the transaction (otherwise), and notifies the result to all the cohorts. The cohorts then follow with the needed actions (commit or abort) with their local transactional resources (also called recoverable resources; e.g., database data) and their respective portions in the transaction's other output (if applicable).

Note that the two-phase commit (2PC) protocol should not be confused with the [two-phase locking](https://en.wikipedia.org/wiki/Two-phase_locking) (2PL) protocol, a [concurrency control](https://en.wikipedia.org/wiki/Concurrency_control) protocol.

The protocol works in the following manner: one node is designated the coordinator, which is the master site, and the rest of the nodes in the network are designated the cohorts. The protocol assumes that there is [stable storage](https://en.wikipedia.org/wiki/Stable_storage) at each node with a [write-ahead log](https://en.wikipedia.org/wiki/Write_ahead_logging), that no node crashes forever, that the data in the write-ahead log is never lost or corrupted in a crash, and that any two nodes can communicate with each other. The last assumption is not too restrictive, as network communication can typically be rerouted. The first two assumptions are much stronger; if a node is totally destroyed then data can be lost.

The protocol is initiated by the coordinator after the last step of the transaction has been reached. The cohorts then respond with an agreement message or an abort message depending on whether the transaction has been processed successfully at the cohort.

## Basic algorithm

### Commit request phase or voting phase

1. The coordinator sends a query to commit message to all cohorts and waits until it has received a reply from all cohorts.
2. The cohorts execute the transaction up to the point where they will be asked to commit. They each write an entry to their undo log and an entry to their [redo log](https://en.wikipedia.org/wiki/Redo_log).
3. Each cohort replies with an agreement message (cohort votes Yes to commit), if the cohort's actions succeeded, or an abort message (cohort votes No, not to commit), if the cohort experiences a failure that will make it impossible to commit.

### Commit phase or Completion phase(Success).

If the coordinator received an agreement message from all cohorts during the commit-request phase:

1. The coordinator sends a commit message to all the cohorts.
2. Each cohort completes the operation, and releases all the locks and resources held during the transaction.
3. Each cohort sends an acknowledgment to the coordinator.
4. The coordinator completes the transaction when all acknowledgments have been received.

#### Failure

If any cohort votes No during the commit-request phase (or the coordinator's timeout expires):

1. The coordinator sends a rollback message to all the cohorts.
2. Each cohort undoes the transaction using the undo log, and releases the resources and locks held during the transaction.
3. Each cohort sends an acknowledgement to the coordinator.
4. The coordinator undoes the transaction when all acknowledgements have been received.

#### Message flow

Coordinator Cohort

QUERY TO COMMIT

-------------------------------->

VOTE YES/NO prepare\*/abort\*

<-------------------------------

commit\*/abort\* COMMIT/ROLLBACK

-------------------------------->

ACKNOWLEDGMENT commit\*/abort\*

<--------------------------------

End

**Viva Questions**

1. Discuss various concurrency control schemes used in Distributed Databases

2. Explain socket programming in JAVA

3. Explain how two phase commit protocol handle the failures.

4. What is multi-threading? How it is used for handling multiple client-threads.(Write a step by step explanation of each method used in multi-threading)