Transactions

Overview

- Transactions
- Transaction Recovery
  - Write Ahead Log [WAL] Rule
  - ACID Properties
- System Recovery
  - Checkpoint
  - Two Phase Commit [2PC]

Recovery

- Recovery in a database system means:

  restoring the database to a correct state after some failure has rendered the current state incorrect.

Transactions

- A transaction is a logical unit of work:

  it begins with the execution of a BEGIN TRANSACTION

  ends with the execution of a COMMIT or ROLLBACK

```
BEGIN TRANSACTION
UPDATE ACC123 (BALANCE := BALANCE - $100);
IF any error THEN GO TO UNDO; END IF;

UPDATE ACC456 (BALANCE := BALANCE + $100);
IF any error THEN GO TO UNDO; END IF;

COMMIT;                    /* successful termination */
GO TO FINISH;

UNDO:    ROLLBACK;         /* unsuccessful termination */
FINISH:  RETURN;
```

Transactions

- If failure occurs before the transaction reaches its planned termination, *then those updates will be undone*.
- The transaction either executes in its entirety or is totally canceled.
- The transaction manager or transaction processing monitor [TP monitor] provides this atomicity.

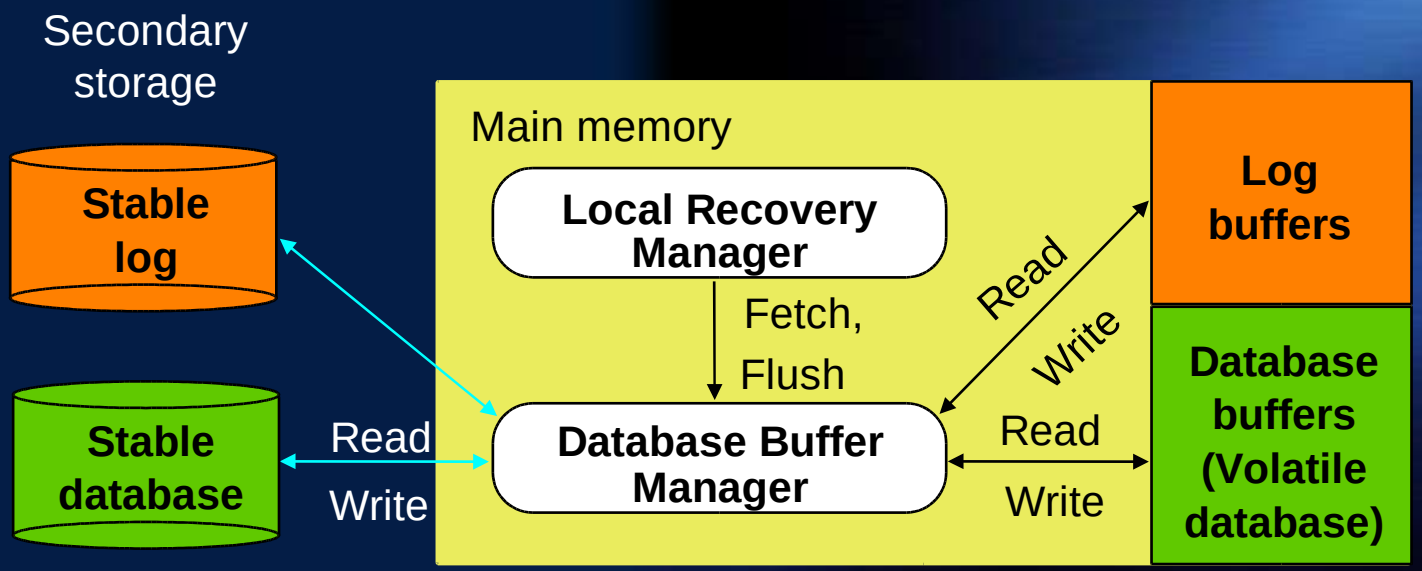
Transactions

- The commit operation signals *successful* end-of-transaction:
  - the database is in a correct state
  - all the updates made by the transaction can be committed
- The rollback operation signals *unsuccessful* end-of-transaction:
  - the database might be in an incorrect state
  - all the updates made by the transaction must be undone
- Recovery log:
  - Log file on which details of all updates – before and after each update – are recorded.

Transaction Recovery

- When a commit point is established:
  - All database updates since the previous commit point are committed – recorded permanent change in the database.
  - All database positioning is lost and all tuple locks are released
    - Database positioning – addressability to certain tuples in database
- Transactions are also the unit of recovery.
- The system might crash after the COMMIT has been honored but before the updates to the database – so lost at the time of crash.

Transaction Recovery



Write Ahead Log [WAL] rule

- Log must be physically written before COMMIT can complete – *write ahead log rule.*
- The log record for a given database update must be physically written to the log before that update is written to the database.
- All other log records for a given transaction must be written to the log before the COMMIT log record for that transaction is written to the log.
- COMMIT processing must not complete until the COMMIT log record for that transaction is written to the log.

The ACID Properties

- Transaction possess the ACID properties –
- **Atomicity** : Transactions are atomic (all or nothing)
- **Correctness** : Transactions transforms a correct state of the database into another correct state
- **Isolation** : Transactions are isolated from one another.

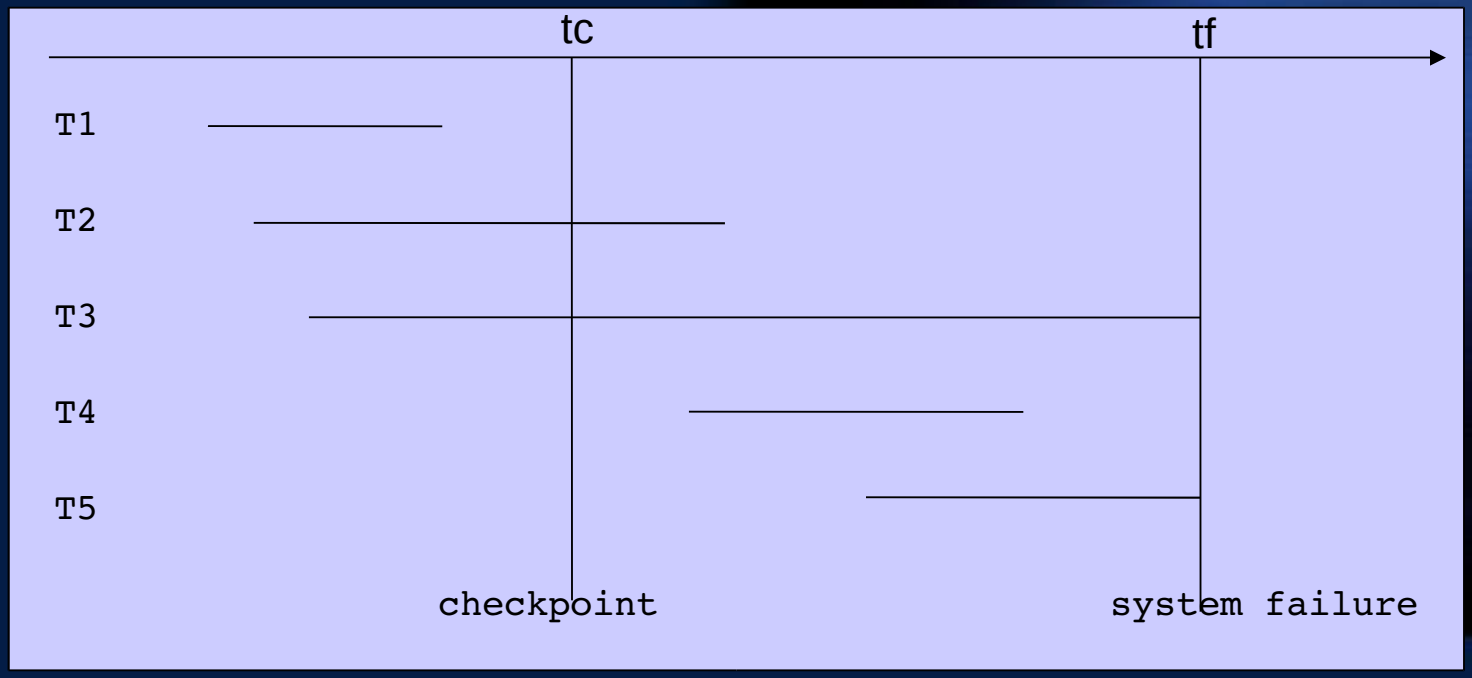
  For any two distinct transactions A and B, A might see B's updates (after B has committed) or B might see A's updates (after A has committed)
- **Durability** : Once a transaction commits, its update persist in the database, even if there is a subsequent system crash

System Recovery

- Local failure – an overflow exception within an individual transaction – affects only the transaction in which the failure has actually occured.
- Global failure – power outage – affects all of the transactions in progress at the time of the failure.
- Two categories of global failures:
- **System failures** – affect all transactions currently in progress but do not physically damage the database – *soft crash – power outage*
- **Media failures** – do cause damage to the database or some portion thereof, and affect at least those transactions currently using that portion – *hard crash – head crash on the disk*

Checkpoint

- In system failure, the contents of main memory are lost.
- The precise state of any transaction that was in progress at the time of the failure is no longer known.



Checkpoint

- When the system restarts, roll back the transactions that was in progress at the time of failure – must be *undone.*
- Redo certain transactions that did successfully complete prior to the crash but did not manage to get their updates transferred from main memory to the physical database.

```
How does the system know at restart time which transactions
to undo and which to redo?
```

- At certain prescribed intervals, the system automatically takes a *checkpoint*

Checkpoint

- Checkpoint involves:
  - Forcing the contents of the main-memory buffers out to the database
  - Forcing a special *checkpoint record* out to the physical log
- At restart time, each transaction are identified as follow:

```
1. Start with two lists of transactions, the UNDO list and the
REDO list.
2. Set UNDO = list of all transactions given in the most recent
checkpoint record; REDO = empty.
3. Search forward through log, starting from checkpoint record.
4. For a transaction T, if a BEGIN TRANSACTION log record is
found, add T to the UNDO list.
```

Checkpoint

- - cont..

```
5. For a transcation T, if a COMMIT log record is found, move T
from UNDO to REDO list.

6. At the end of the log, the UNDO and REDO lists identify,

transactions T3 and T5 has to be UNDONE and

transactions T2 and T4 has to be REDONE.
```

- UNDO – work backward through the log – backward recovery
- REDO – work forward through the log – forward recovery

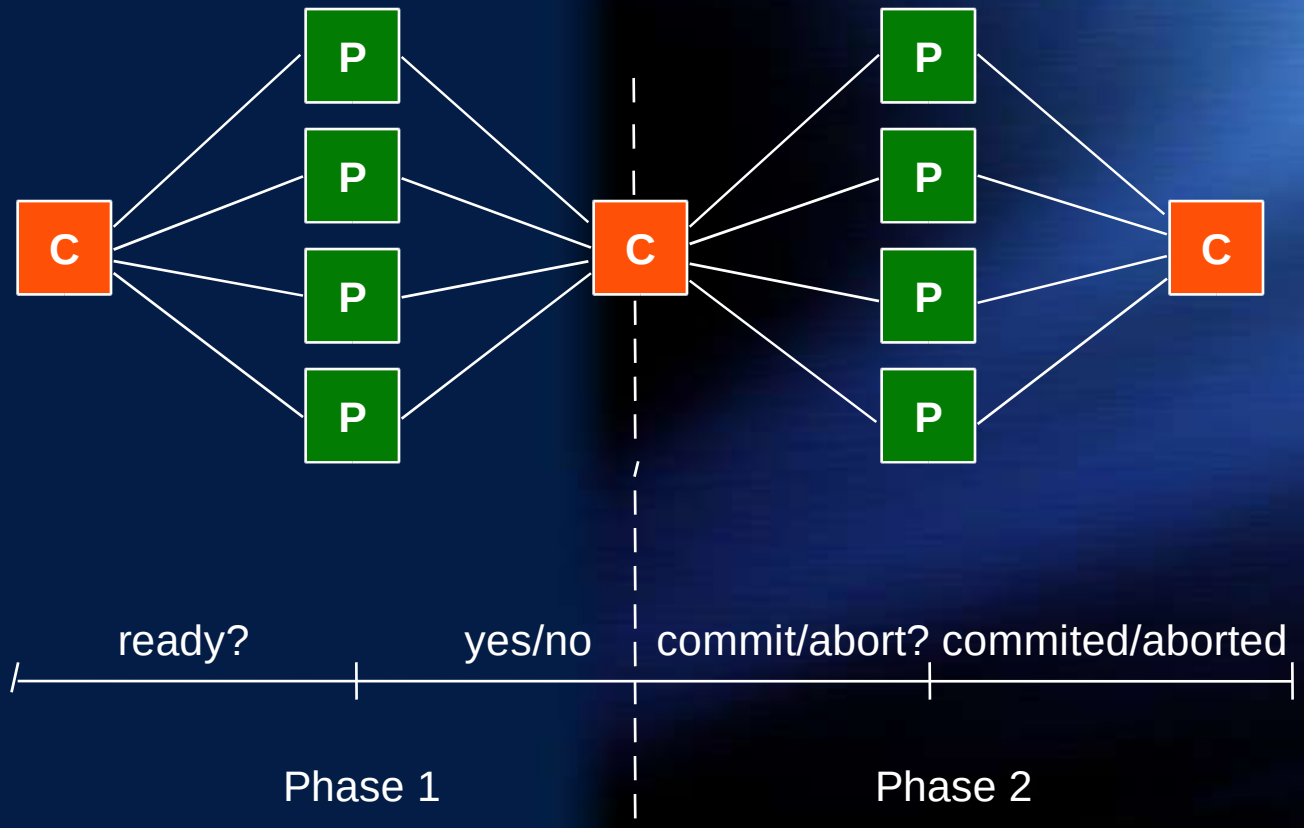
Two-Phase Commit [2PC]

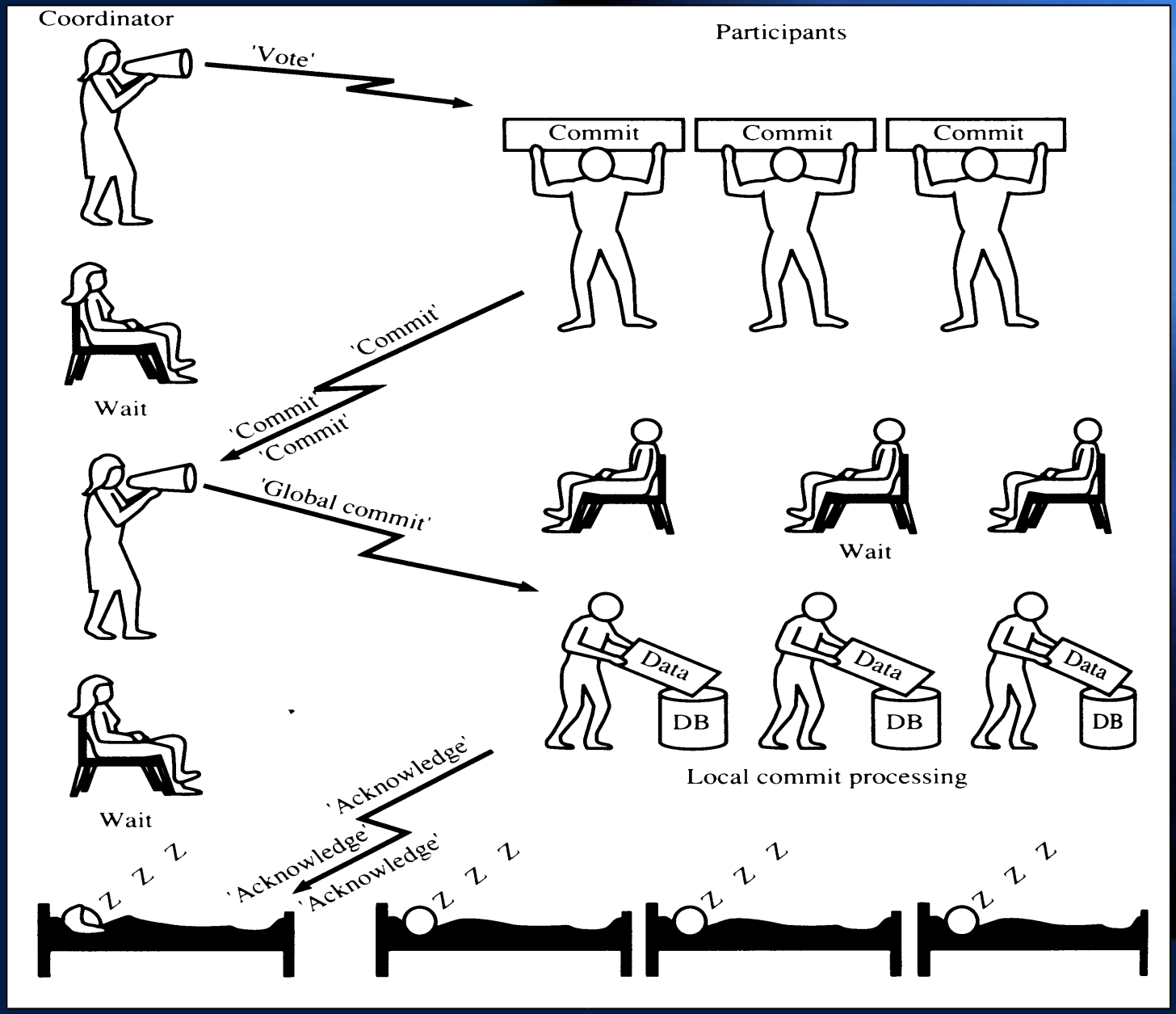
- The two-phase commit protocol ensures that all participating resources (database servers) receive and implement the same action (either to commit or to roll back a transaction).
- Every global transaction has a coordinator and one or more participants, defined as follows:
  - The coordinator directs the resolution of the global transaction. It decides whether the global transaction should be committed or stopped.
  - The two-phase commit protocol always assigns the role of coordinator to the current database server. The role of coordinator cannot change during a single transaction.

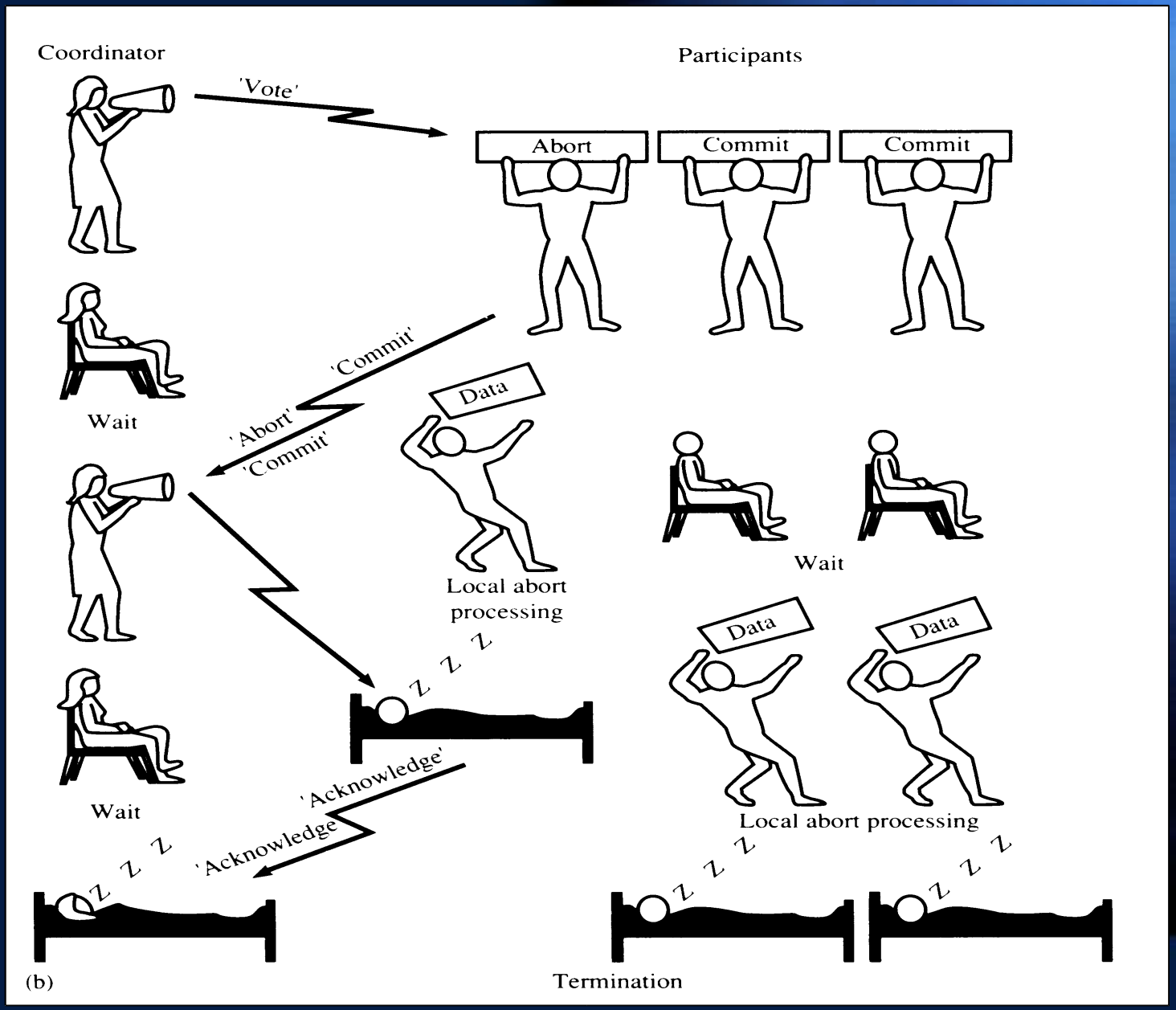
Two-Phase Commit [2PC]

- PHASE – I [Prepare]
  - The coordinator instructs all participants to get ready on the transaction – must force all log records out to its own physical log.
  - If forced write is successful, the participants now replies 'YES' to the coordinator; otherwise 'NO'.
- PHASE – II [Commit]
  - Coordinator forces a record to its own physical log, recording its decision.
  - If all replies were 'YES', decision is 'COMMIT'; if any reply was 'NO', the decision is 'ROLLBACK'.
  - Coordinator informs each participant of its decision, and each participant must then commit / rollback the transaction locally, as instructed.

Two-Phase Commit [2PC]




Coordinator
Participants
'Vote'
Commit
Commit
Commit
Wait
'Commit'
'Commit'
'Commit'
'Global commit'
Wait
Data
DB
Data
DB
Data
DB
Local commit processing
Wait
'Acknowledge'
'Acknowledge'
'Acknowledge'
Z Z Z
Z Z Z
Z Z Z
Z Z Z


Coordinator
Participants
'Vote'
Abort
Commit
Commit
'Commit'
'Abort'
'Commit'
Wait
Data
Local abort processing
Wait
Data
Data
Wait
'Acknowledge'
'Acknowledge'
Local abort processing
(b)
Termination

References

- *Chapter 15: Recovery*
  An introduction to database systems, *CJ. Date*