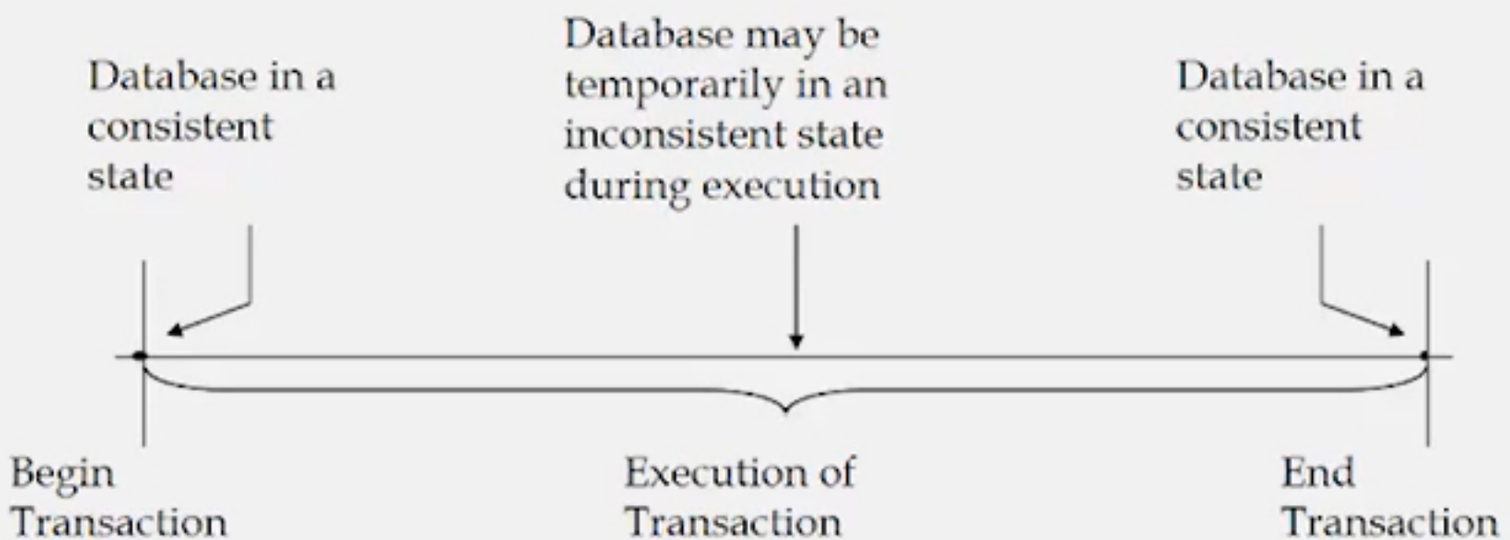
Introduction to Database System

Recovery

Ira A. Fulton Schools of Engineering
Arizona State University

Transaction

A transaction is a collection of actions that make consistent transformations of system states while preserving system consistency.



Principles of Transactions

ATOMICITY

- all or nothing

CONSISTENCY

- no violation of integrity constraints

ISOLATION

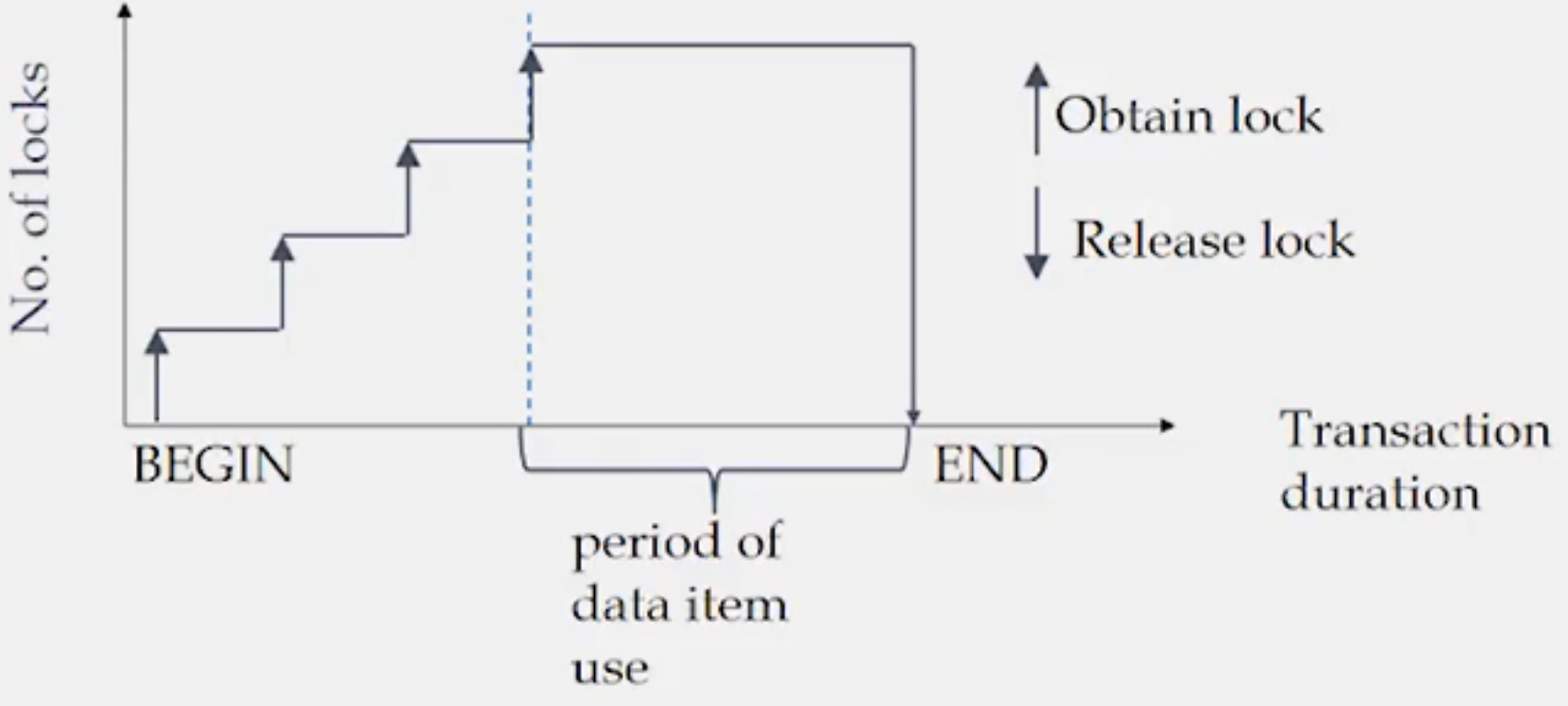
- concurrent changes invisible $\Rightarrow$ serializable

DURABILITY

- committed updates persist

Strict 2PL

Hold locks until the end.



Deadlocks

- Deadlock: Cycle of transactions waiting for locks to be released by each other.
- Two ways of dealing with deadlocks:
  - Deadlock prevention
  - Deadlock detection

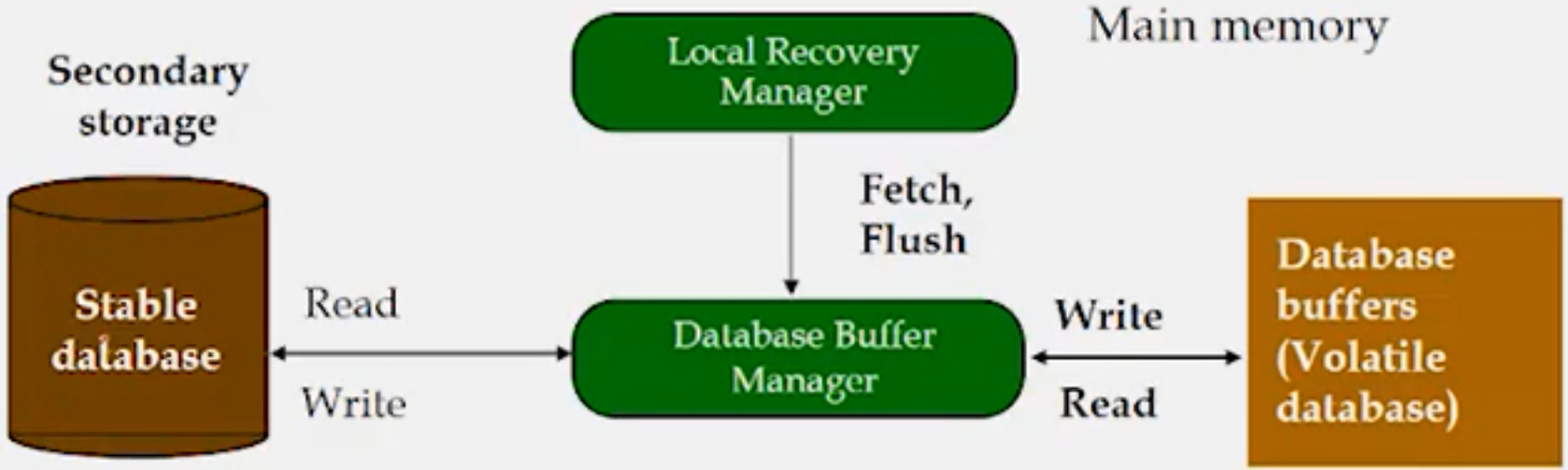
Faults to Failures

Types of Failures

- Transaction failures
  - Transaction aborts (unilaterally or due to deadlock)
  - Avg. 3% of transactions abort abnormally
- System failures
  - Failure of processor, main memory, power supply, …
  - Main memory contents are lost, but secondary storage contents are safe
  - Partial vs. total failure
- Media failures
  - Failure of secondary storage devices such that the stored data is lost
  - Head crash/controller failure (?)
- Communication failures
  - Lost/undeliverable messages
  - Network partitioning

Local Recovery Management – Architecture

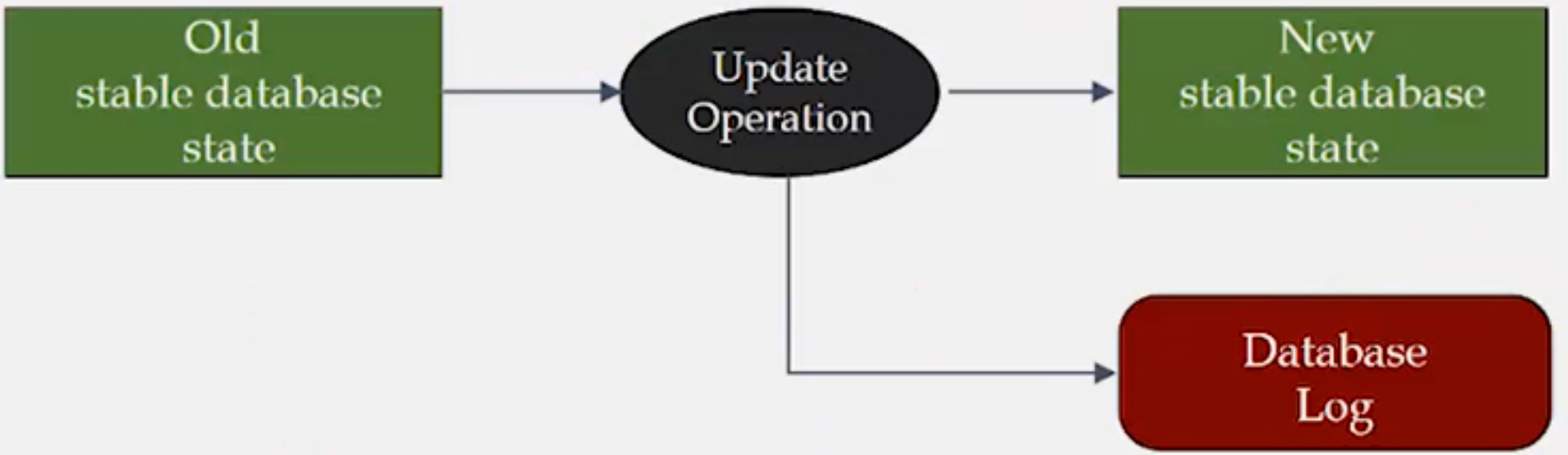
- Volatile storage
  - Consists of the main memory of the computer system (RAM).
- Stable/Persistent storage
  - Resilient to failures and loses its contents only in the presence of media failures (e.g., head crashes on disks).
  - Implemented via a combination of hardware (non-volatile storage) and software (stable-write, stable-read, clean-up) components.



In-Place Update Recovery Information

Database Log

Every action of a transaction must not only perform the action, but must also write a *log* record to an append-only file.



The Log

- The following actions are recorded in the log:
  - *Ti writes an object*: the old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts*: a log record indicating this action.
- Log records are chained together by Xact id, so it's easy to undo a specific Xact.
- All log related activities are handled transparently by the DBMS.

Logging

The log contains information used by the recovery process to restore the consistency of a system. This information may include

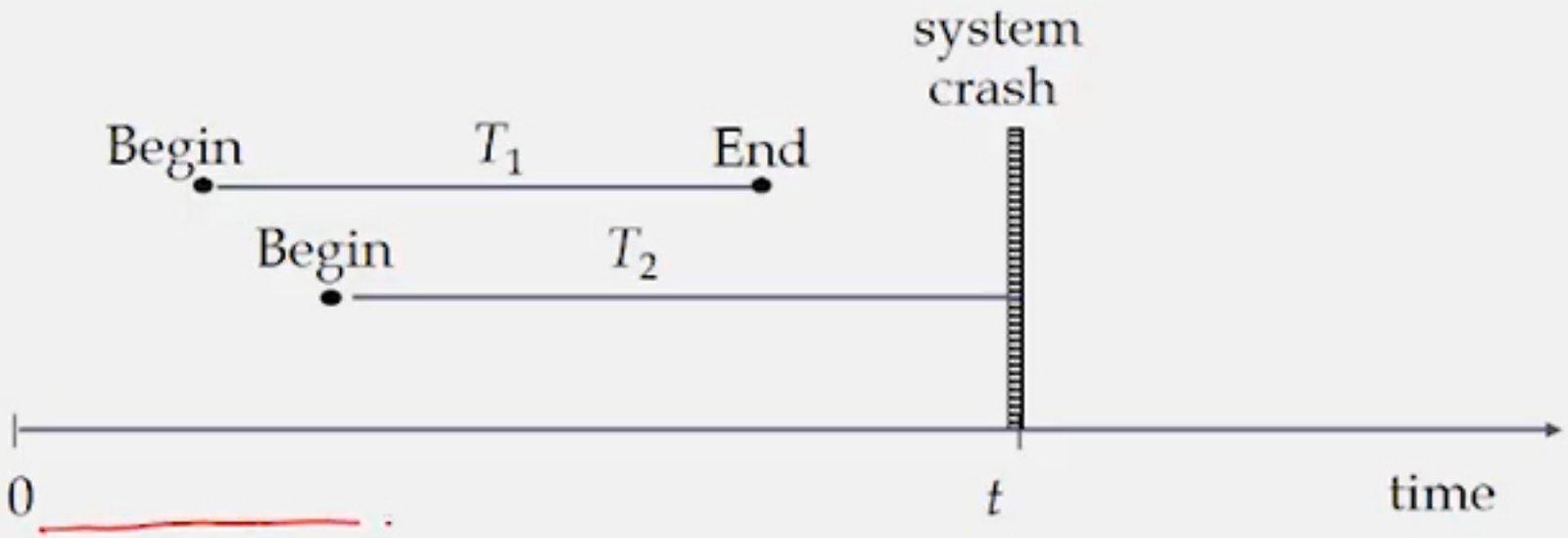
- transaction identifier
- type of operation (action)
- items accessed by the transaction to perform the action
- old value (state) of item (before image)
- new value (state) of item (after image)

...

Why Logging?

Upon recovery:

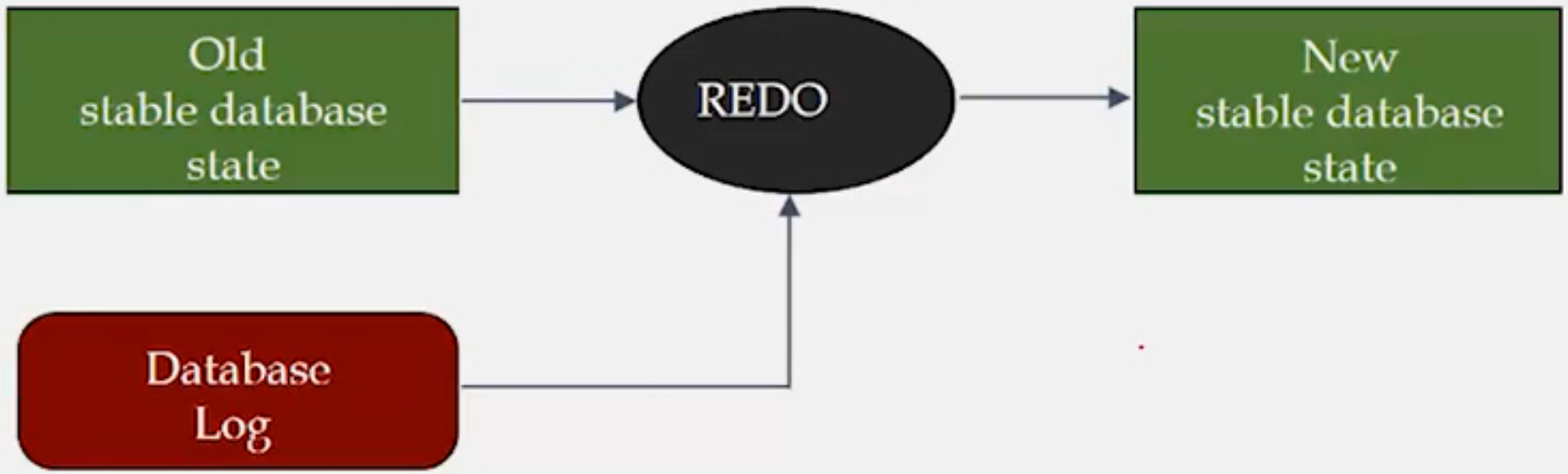
- all of T_1's effects should be reflected in the database (REDO if necessary due to a failure)
- none of T_2's effects should be reflected in the database (UNDO if necessary)



Write–Ahead Log Protocol

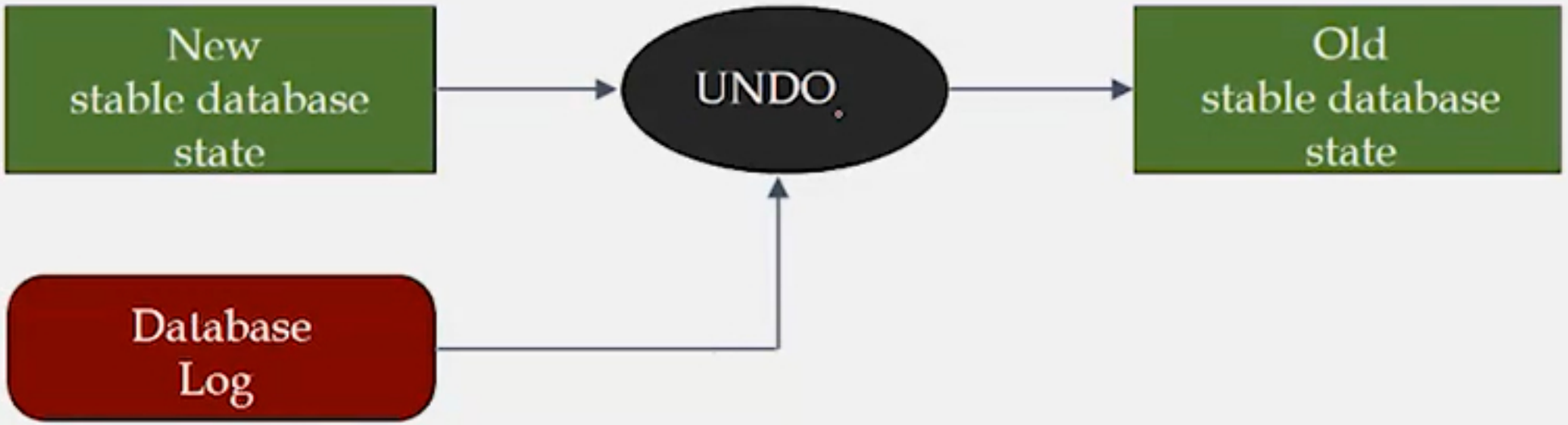
- Notice:
  - If a system crashes before a transaction is committed, then all the operations must be undone. Only need the before images (*undo portion* of the log).
  - Once a transaction is committed, some of its actions might have to be redone. Need the after images (*redo portion* of the log).
- WAL protocol :
  1. Before a stable database is updated, the undo portion of the log should be written to the stable log
  2. When a transaction commits, the redo portion of the log must be written to stable log prior to the updating of the stable database.

REDO Protocol



- REDO'ing an action means performing it again.
- The REDO operation uses the log information and performs the action that might have been done before, or not done due to failures.
- The REDO operation generates the new image.

UNDO Protocol



- UNDO'ing an action means to restore the object to its before image.
- The UNDO operation uses the log information and restores the old value of the object.

Summary

- Concurrency control and recovery are among the most important functions provided by a DBMS.
- Users need not worry about concurrency.
  - System automatically inserts lock/unlock requests and schedules actions of different Xacts in such a way as to ensure that the resulting execution is equivalent to executing the Xacts one after the other in some order.
- Write-ahead logging (WAL) is used to undo the actions of aborted transactions and to restore the system to a consistent state after a crash.
  - *Consistent state*: Only the effects of commited Xacts seen.