SHADOW PAGING

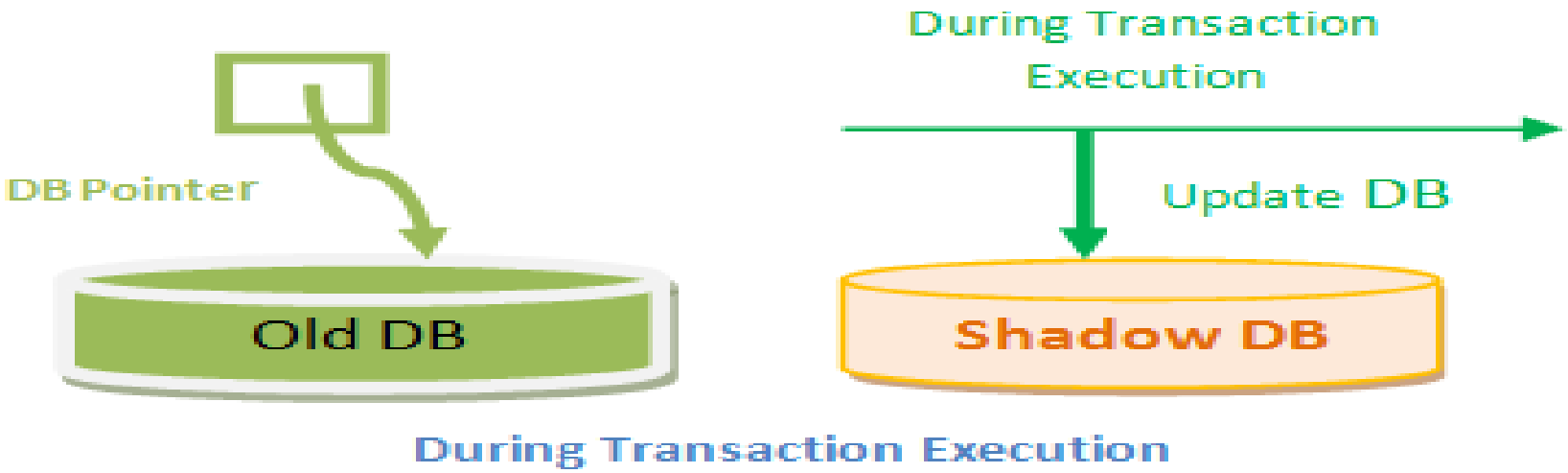
SHADOW PAGING

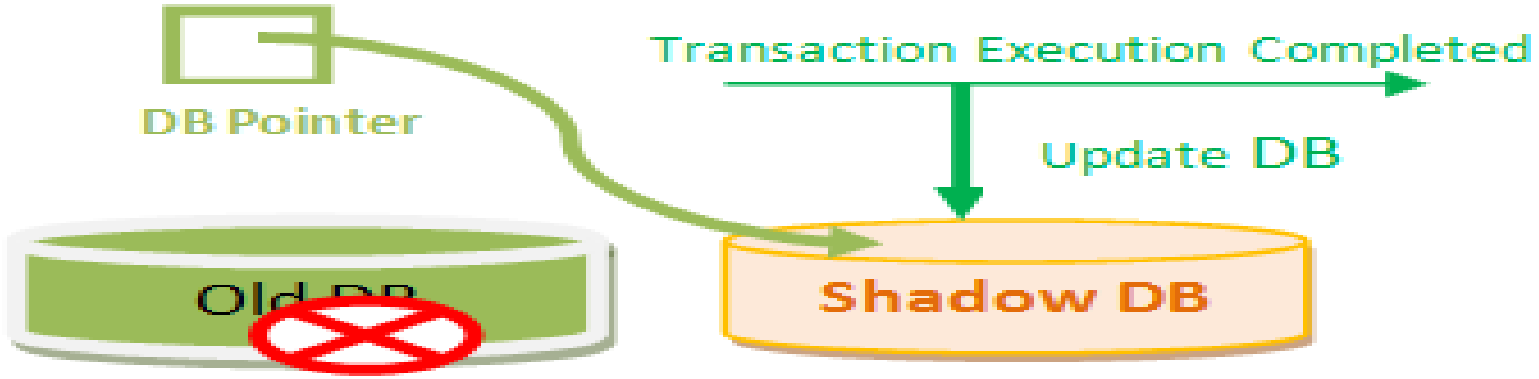
- Shadow paging is a technique for providing atomicity and durability in database systems.
- Shadow paging is a copy-on-write technique for avoiding in-place updates of pages. Instead, when a page is to be modified, a shadow page is allocated.
- Since the shadow page has no references (from other pages on disk), it can be modified liberally, without concern for consistency constraints, etc.
- When the page is ready to become durable, all pages that referred to the original are updated to refer to the new replacement page instead. Because the page is "activated" only when it is ready, it is atomic.

SHADOW PAGING

- This increases performance significantly by avoiding many writes on hotspots high up in the referential hierarchy (e.g.: a file system superblock) at the cost of high commit latency.
- This is the method where all the transactions are executed in the primary memory or the shadow copy of database.
- Once all the transactions completely executed, it will be updated to the database.
- Hence, if there is any failure in the middle of transaction, it will not be reflected in the database.
- Database will be updated after all the transaction is complete.

SHADOW PAGING





SHADOW PAGING

- A database pointer will be always pointing to the consistent copy of the database, and copy of the database is used by transactions to update.
- Once all the transactions are complete, the DB pointer is modified to point to new copy of DB, and old copy is deleted.
- If there is any failure during the transaction, the pointer will be still pointing to old copy of database, and shadow database will be deleted.
- If the transactions are complete then the pointer is changed to point to shadow DB, and old DB is deleted.

SHADOW PAGING

Shadow paging considers:

- The database is partitioned into fixed-length blocks referred to as PAGES.
- Page table has n entries – one for each database page.
- Each contain pointer to a page on disk (1 to 1st page on database and so on…).

The idea is to maintain 2 pages tables during the life of transaction.

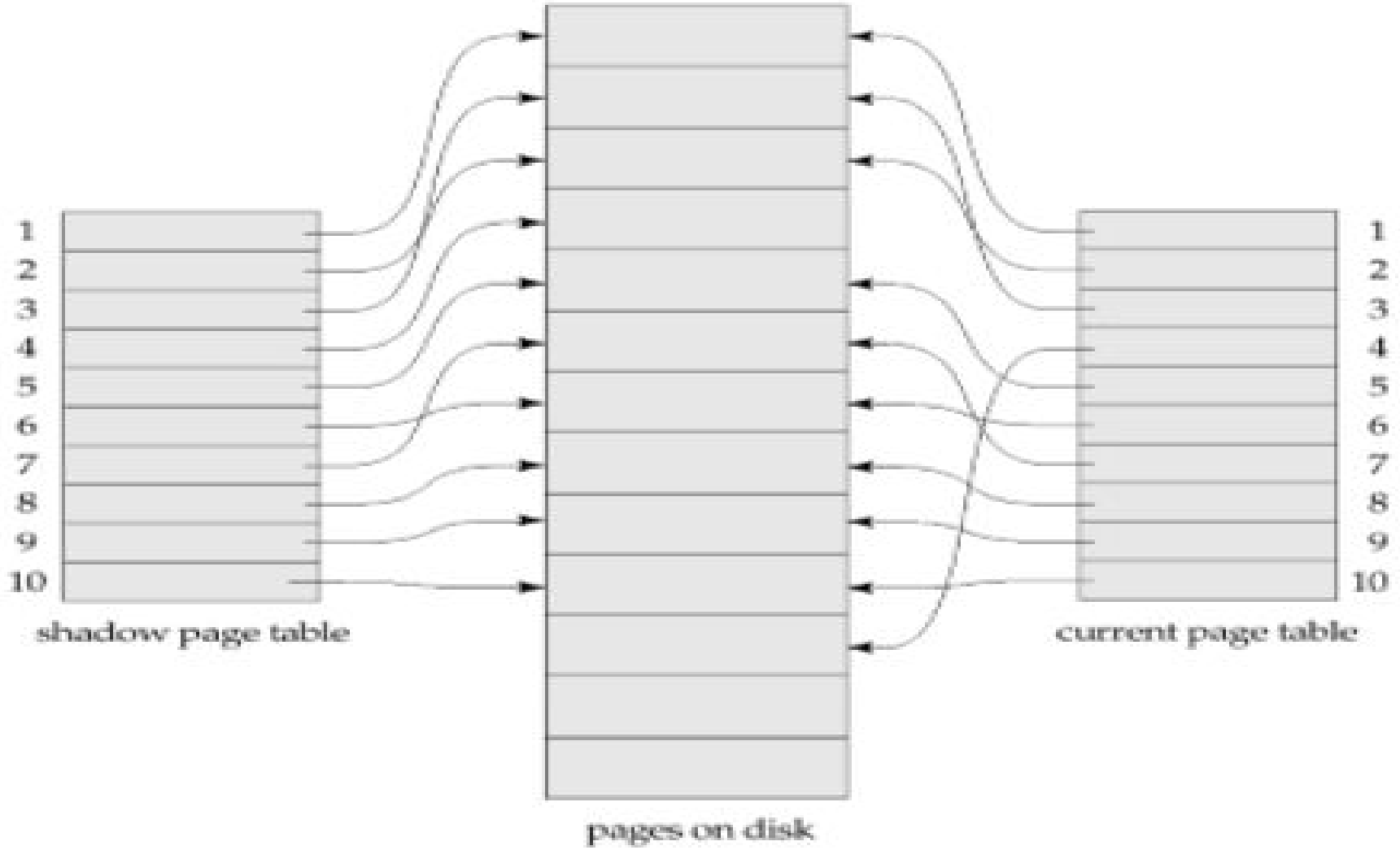
- The current page table - Current page table points to most recent current database
- The shadow page table

SHADOW PAGING

When transaction starts executing,

- Both page tables are identical i.e., The Current page table is copied into a shadow page table
- The shadow page table is never changed over the duration of the transaction.
- The current page table may be changed when a transaction performs a write operation.
- All input and output operations use the current page table to locate database pages on disk.

SHADOW PAGING



SHADOW PAGING

To recover from a failure:

- The state of the database before transaction execution is available through the shadow page table
- Free modified pages
- Discard current page table
- That state is recovered by reinstating the shadow page table to become the current page table once more.
- Committing a transaction
- Discard previous shadow page
- Free old page tables that it references

SHADOW PAGING

Advantages:

- No Overhead for writing log records.
- No Undo / No Redo algorithm.
- Recovery is faster.

Disadvantages:

- Data gets fragmented or scattered.
- After every transaction completion database pages containing old version of modified data need to be garbage collected.
- Hard to extend algorithm to allow transaction to run concurrently.