Chapter 3

Part 1 DATABASE RECOVERY TECHNIQUES

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Outline

1. Goals
2. Type of Failures
3. Major Concepts
    - Transaction log
    - Checkpoint
4. Database Recovery Techniques

Example

- Accounts: A : 1000 $, B : 2000 $
- Transfer 50 $ from A to B.
- Let's consider the state:
  - A had updated: A:= A-50
  - B had not been updated: B:= B+50
  - And there was power cut!
- When the system restarted:
  - If T was executed again: A = 900
  - If not: A = 950 and B =2000.
  - How should the system be recovered?

Goals

- Database Recovery is the process of restoring the database and the data to a consistent state. This may include restoring lost data up to the point of the event (e.g. system crash).
- This is done by RM – the Recovery Manager.
- Automatic database recovery helps saving manual effort after every crash.

Goals

- Transaction is the basic unit of Database Recovery.
- Among ACID properties of Transaction, Database Recovery helps retain Atomic and Durability.

Types of failures

- Transaction failures
  - Rolling back caused by Deadlock or as requested by the scheduler.
  - During failure, system still operates normally.
  - Frequency: some times/minute.
- System failures
  - System can nolonger operate. The reason may be failures in the processing unit, power cut or software failure.
  - Only cause data lost on RAM.
  - Frequency: some times/month.

Types of failures

- Media failure
  - Ex̣: HDD crash.
  - Cause the lost of a part or entire database.
  - Frequency: some times/year.
- Software failures
  - Logical error of applications which access database, leading to transaction failure.

Note

□ Regardless of failure reasons, we have to consider 2 issues:

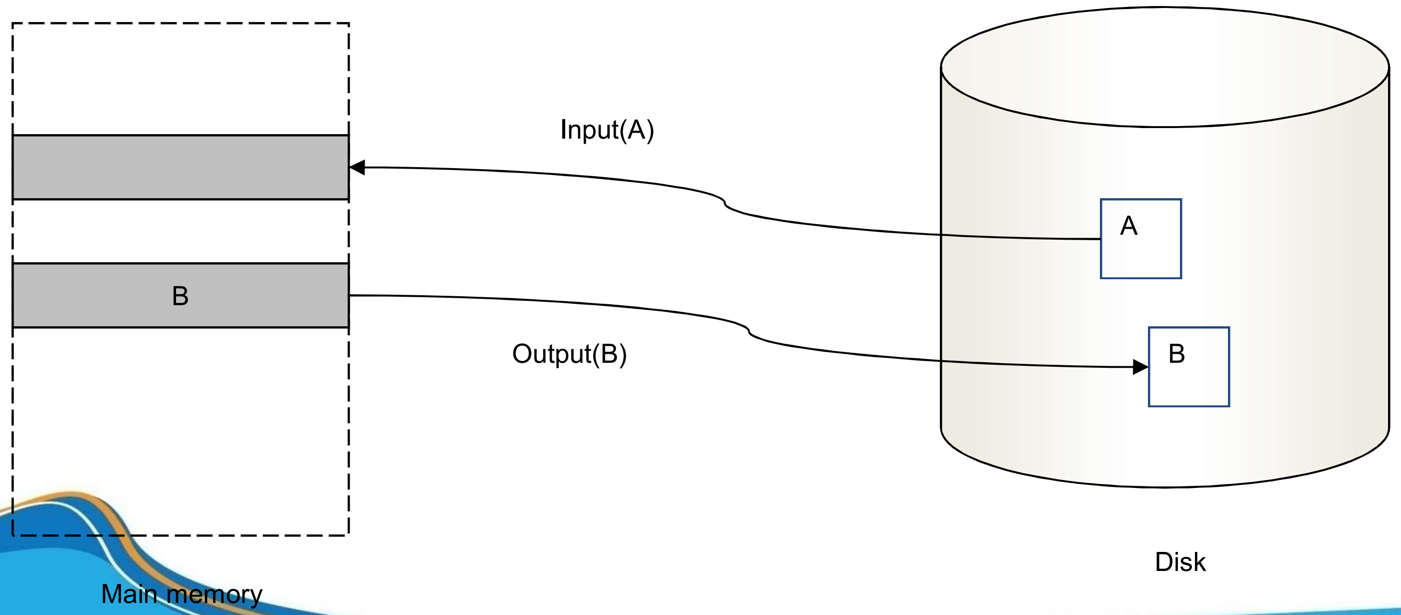
1. Data lost on database buffer.
2. Data lost on storage media.

Backup

- DBMS provide Backup mechanism.
  - Full backup / Differential backup.
  - Handle data lost on storage media.

DB access

- Data is read or written in primary unit called “block”.
  - Physical block: Stored in Media.
  - Buffer block: Stored in buffer.



DB access

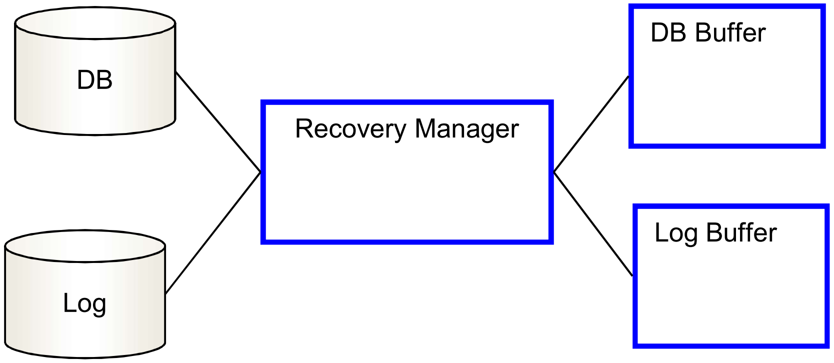
- Read (X): assign X to local variable
  - If the data block with X is not in buffer, then Input (X).
  - assign X to local variable Xi.
- Write (X): assign Xi to X
  - If the data block with X is not in buffer, then Input (X)..
  - Gán giá trị xi cho X (trên buffer block có chứa X).

Buffer management

- Buffer:
  - Dữ liệu mất khi có sự cố hệ thống.
  - Không gian hạn chế.
- Chiến lược thay thế để định ra vùng trống trên buffer dùng để nạp dữ liệu mới.
  - FIFO.
  - LRU.

Note

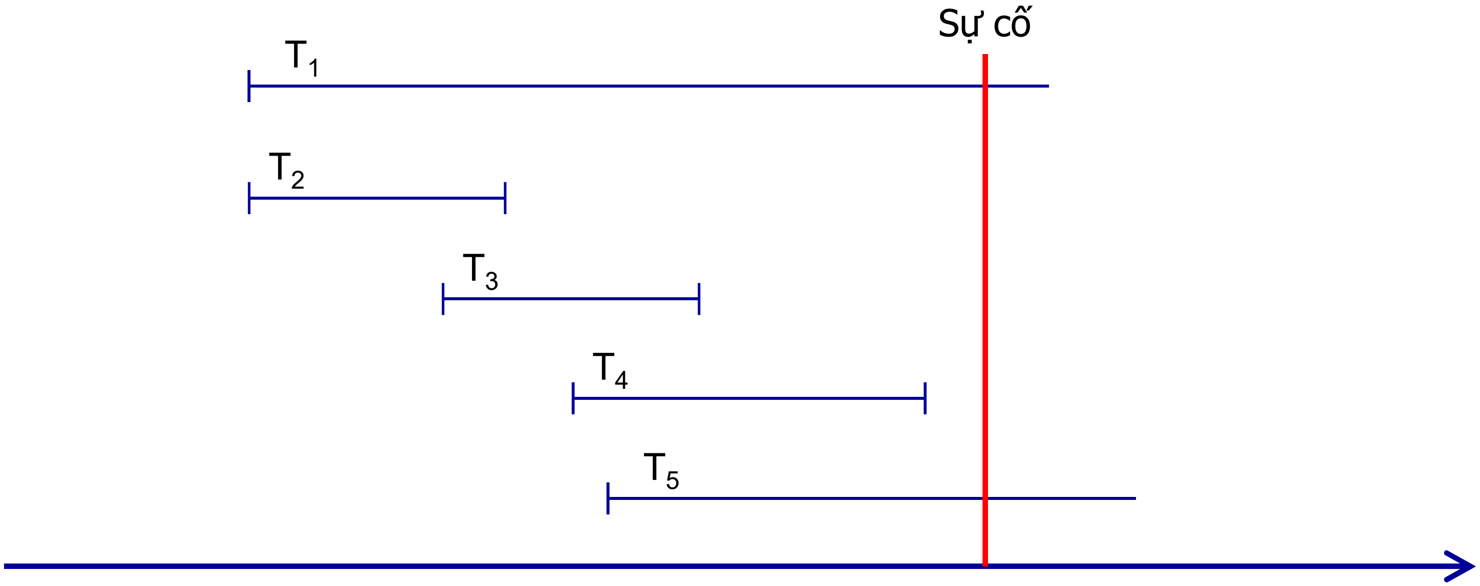
- Database has 2 parts:
  - Physical Database and
  - Database Buffer
- Log has 2 parts:
  - Physical Log and
  - Log Buffer
- Failure ⇒ data lost in database buffer ⇒ use log file to recover data.
- Failure ⇒ data lost in log buffer ⇒ redo DB manipulation which has not been saved to physical DB.



Note

- Data in buffer is flushed to dish in some cases, such as:
  - A particular command (eg. Commit).
  - When buffer get full.
- Implicit flush: force-writing.
- If failure occurs befor data flush:
  - For transactions already committed, RM redo their actions (rollforward) for Durability.
  - For transactions not yet committed, RM undo their actions (rollback) for Atomicity.

Example



When failure occurs, T2 , T3, T4 has committed → RM redo their data manipulation when DB restarts.

T1 and T5 will rollback

- DBMS provides these Recovery utilities:
  - RM.
  - Backup.
  - Logging.
  - Checkpoint.

Steal & No-force

- RM use 2 methods for flushing data from buffer to dish:
  - Steal policy: data in buffer is flushed to disk **Befor transaction commit**. V.s no-steal, flushing nothing befor transaction commit.
  - Force policy: data in buffer is flushed to disk immediately when **Transaction commit**. V.s no-force.
- For no-steal → No need to undo changes by uncommitted transactions.
- For force → No need to redo changes by committed transactions.
- Steal policy helps keep buffer from being full of need-flushing blocks.
- No force is beneficial when 2 transactions access the same block, the latter will not have to reload the block from disk.
- Most DBMS use: steal, no-force.

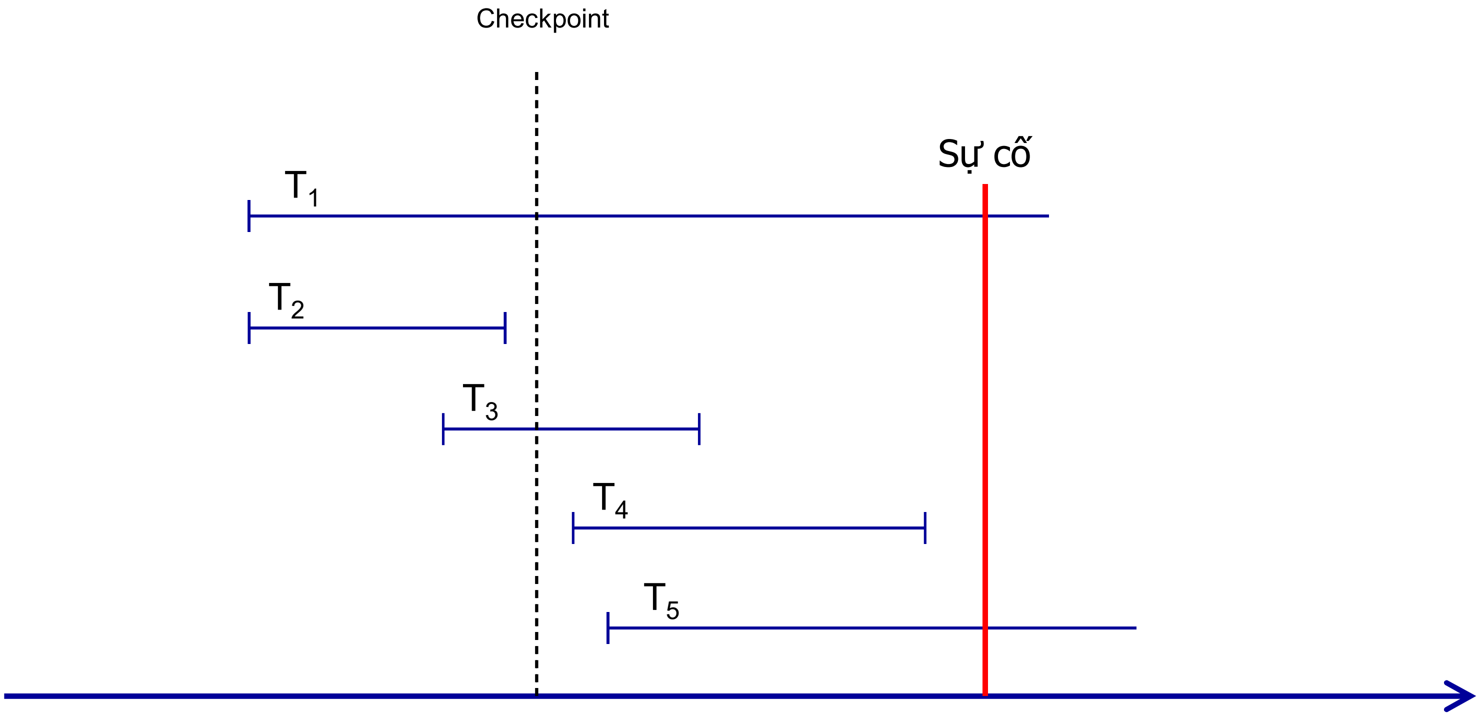
Log file

- Log file audits all changes made to DB.
- Log file is used for DB Recovery, consists of:
  - Transaction record:
    1. ID of transactions.
    2. Log records(Transaction start, insert, update, delete, abort, commit)
    3. ID of data items.
    4. Data items' old values ⇔ Before Image
    5. Data items' new values ⇔ After Image
    6. Pointer managing log records in log file.
  - Checkpoint record.
- Because of log files' extreme importance, DBMS often maintains 2 or 3 copy of log files simultaneously.

Checkpoint

- Recovery using log files has a drawback:
  - Scan the entire log → time consuming.
  - Unnecessary process of log records for already-written data blocks.
  - Checkpoint is introduced to address this disadvantage.
- Checkpoint will periodically:
  - Flush to log files all un-written log records in Log buffer.
  - Flush to disk all un-written data blocks in DB buffer.
  - Write checkpoint record to log file.
- RM decides the interval for checkpoint, may be after n minutes or when t transactions have commited after the previous checkpoint.

Checkpoint example



T2 has been flushed, no need to redo T2.

Recovery techniques

- Recovery techniques using deferred update
- Recovery techniques using immediate update

Recovery techniques using deferred update

- No flushing until transaction commit. (no-steal.)
- Transaction failure befor commit → No undo.
- Redo committed transactions.
- For transaction T, use log as:
  - Transaction T start, note to log.
  - No changes made to DB buffer or disk during transaction T.
  - Write all log records and commit record for T to log file.
  - Use log records to actually manipulate DB.
  - If T aborts, ignore all its log records, doing nothing to the DB.

Recovery techniques using immediate update

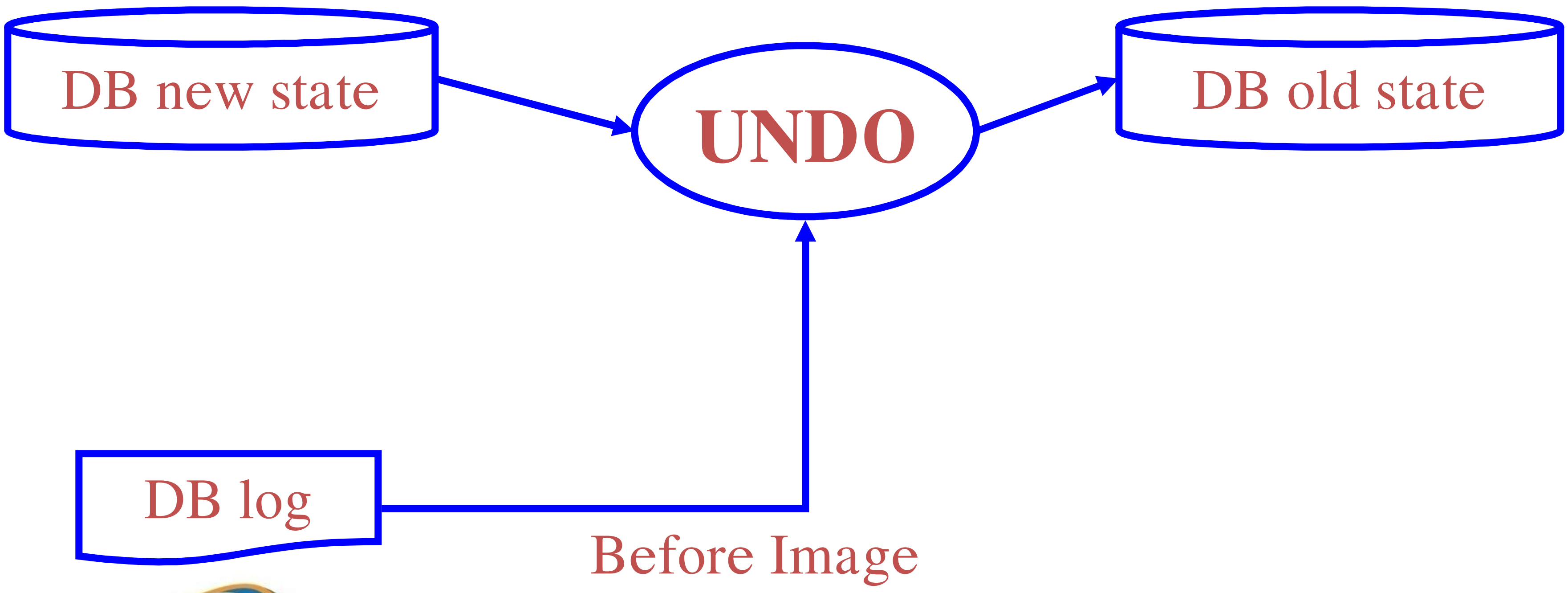
- Execute every transaction command immediately, does not wait until commit.
- When failure occurs:
  - Redo committed transactions.
  - Undo uncommitted transactions.
- For transaction T, use log file as:
  - T starts, note to log filẹ.
  - Write every log records for T to log file.
  - Data changes are stored in DB buffer and flushed to disk when appropriate.
  - If T commit, note to log file.

WAL Protocol (Write Ahead Log)

- When failure, unflushed log records in log buffer may be lost in the same way as unflushed data blocks in DB buffer.
- Log records must be written to log file before corresponding data changes are flushed to DB on disk (Write Ahead Log Protocol).

UNDO protocol

- □ Undo uncommitted transactions or rollbacked transactions



DB new state

UNDO

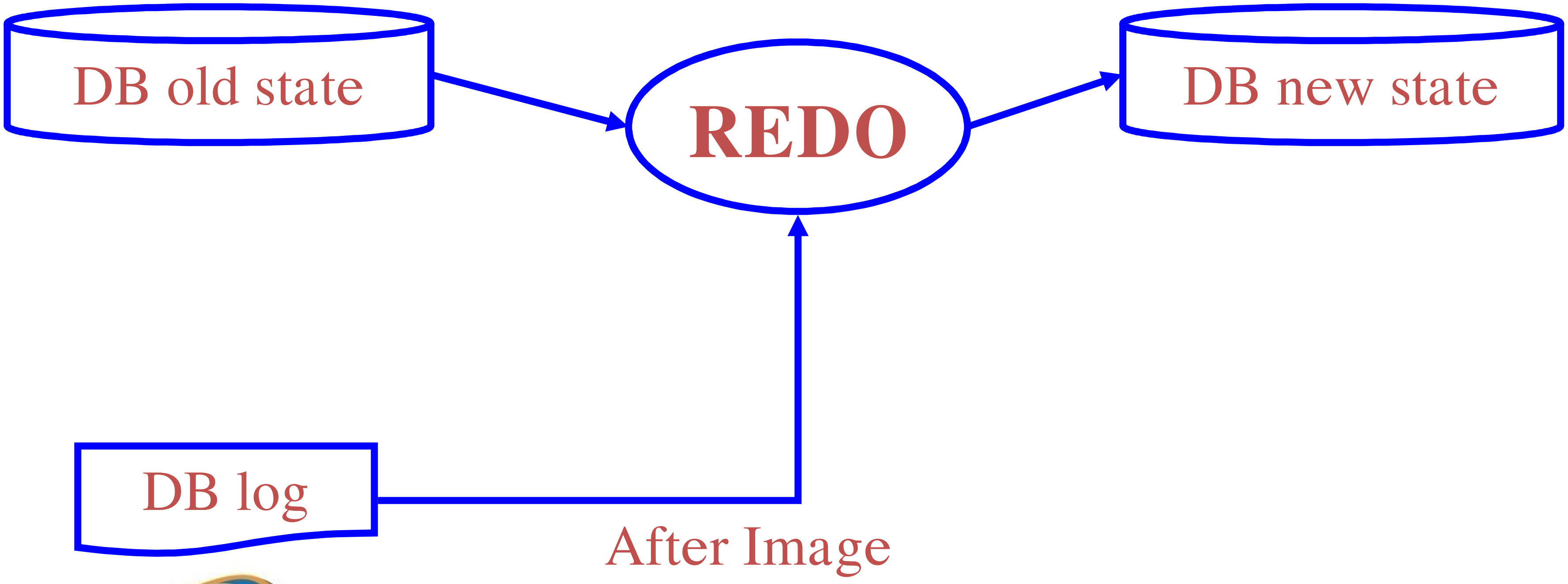
DB old state

DB log

Before Image

REDO Protocol

□ Redo committed transactions which are not yet flushed to disk.



DB old state

REDO

DB new state

DB log

After Image

Normal recovery

- After a normal system shutdown, 1 checkpoint is written to the end of log file.
- When system starts, thanks to this checkpoint at the end of log file, no Undo or Redo is needed .

Failure recovery

- If the last record in log file is checkpoint, STOP.
- Else, find the last checkpoint in log file.
- Indentify 2 transaction group:
  - Group 1: Committed transaction.
  - Group 2:
    - Uncommitted transactions.
    - Rollbacked transactions.
- For group 1: Redo.
- For group 2: Undo.

Convention

- Undo actions with ↑
  - ↑ Need actual undo on DB using before image.
  - [↑] No need actual undo on DB because actions take place after last checkpoint, changes are only in log file.
- Redo actions with ×

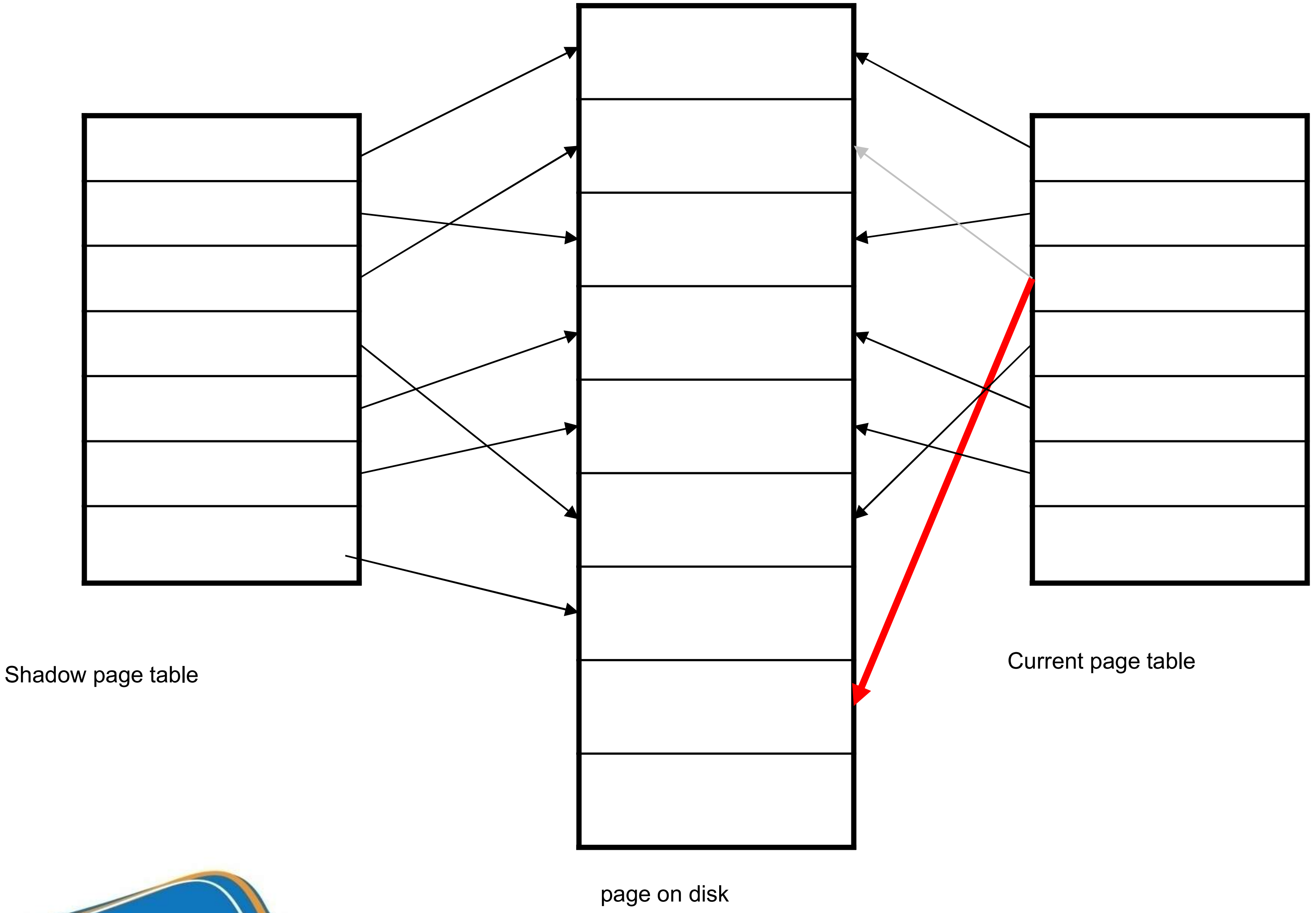
Ex:

BOT$_i$	**Bắt đầu giao tác T$_i$**
U1(i)	**Cập nhật lần 1 của Ti**
BOTi+1	**Bắt đầu giao tác Ti+1**
U1(i+1) ↑	**Thao tác cập nhật thứ 1 của giao tác Ti+1**
Checkpoint	
BOTi+2	**Bắt đầu giao tác Ti+2**
U1(i+2) ×	**Thao tác cập nhật thứ 1 của giao tác Ti+2**
U2(i) ×	**Thao tác cập nhật thứ 2 của giao tác Ti**
Commit Ti	**Commit Ti**
U2(i+1) [↑]	**Thao tác cập nhật thứ 2 của giao tác Ti+1**
BOTi+3	**Bắt đầu giao tác Ti+3**
U1(i+3) [↑]	**Thao tác cập nhật thứ 1 của giao tác Ti+3**
U2(i+3) [↑]	**Thao tác cập nhật thứ 2 của giao tác Ti+3**
U2(i+2) ×	**Thao tác cập nhật thứ 2 của giao tác Ti+2**
Commit Ti+2	**Commit Ti+2**
U3(i+1) [↑]	**Thao tác cập nhật thứ 3 của giao tác Ti+1**
	Sự cố hệ thống xảy ra

Shadow paging

- Another Recovery method is Shadow paging.
- During transaction T's entire life time, 2 tables are maintained:
  - Current page table: Changed as T writes data.
  - Shadow page table: A copy of the table before T starts.
  - When T starts, the 2 page tables are the same.

Shadow paging



End of chapter 3. Part 1