Transaction Management

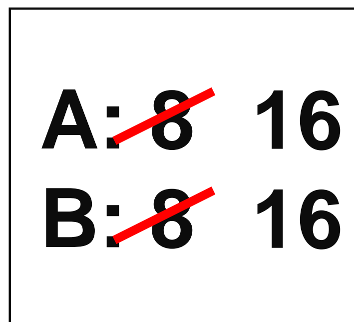
Recovery (2)

Review: Undo Logging

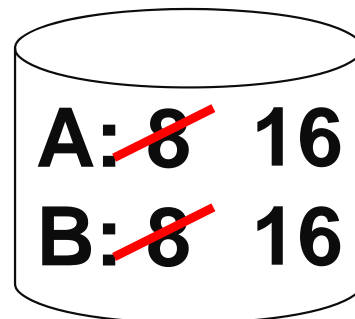
Constraints: A=B

T_1: Read (A,t); $t \leftarrow t \times 2$
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);

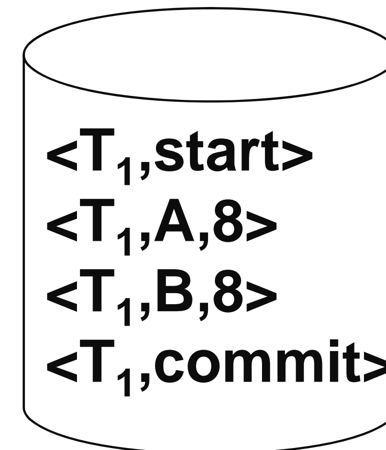
Immediate Modification



Memory



Disk



Undo Log

Disadvantages of Undo Logging

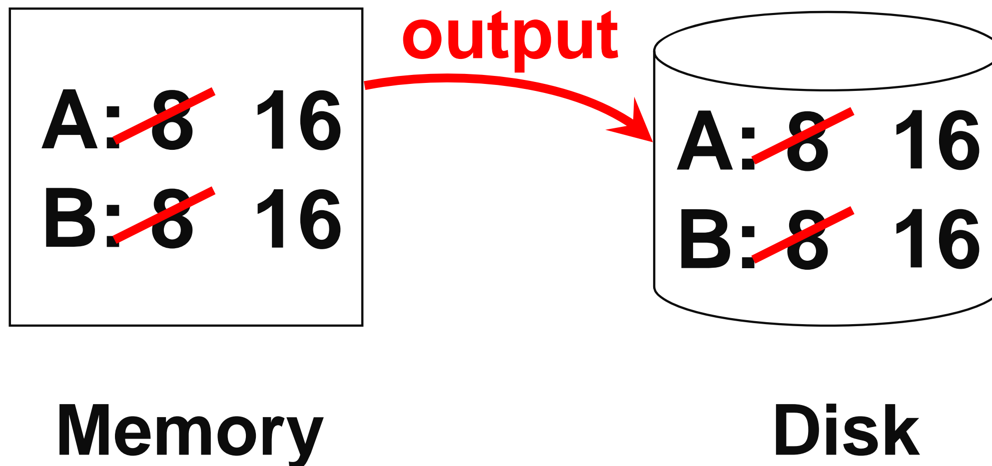
- In Undo Logging, we need to write all modified data to disk before committing a transaction.
- This may require an unnecessarily large number of disk I/O's.
- Can we save disk I/O's if we allow to let changes to data reside in memory for a while?

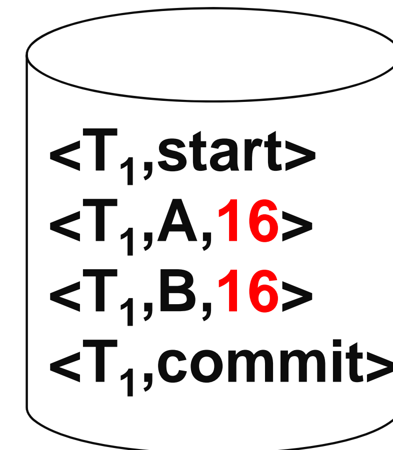
Redo Logging

T_1: Read (A,t); $t \leftarrow t \times 2$
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);

Constraints: A=B

Deferred Modification





Memory

Disk

Redo Log

Redo Logging Rules

(1) For every action, generate redo log record (containing new value).

(2) Before X is modified on disk, all log records for transaction that modifies X (including commit) must be on disk.

(3) Flush log at commit.

Recovery Rules: Redo Logging

- For every T_i with <T_i,commit> in log:
  - For all <T_i,X,v> in log:

    Write(X,v)
    Output(X)

IS THIS CORRECT?

Recovery Rules: Redo Logging

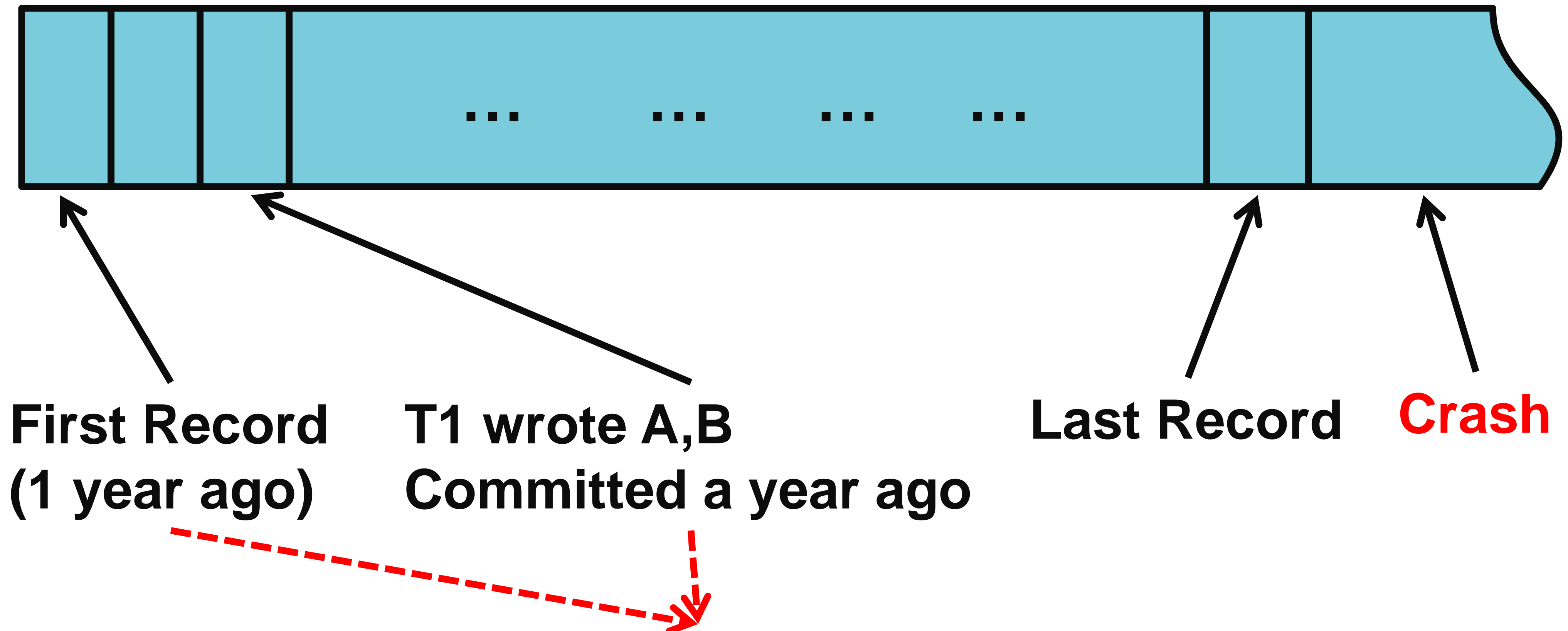
(1) Let S=set of transactions with <T_i,commit> in log;

(2) For each <T_i,X,v> in log, in forward order (earliest → latest) do:

if $T_i \in S$, then

$$\begin{cases} \text{Write}(X,v) \\ \text{Output}(X) \end{cases}$$

Recovery is SLOW!



Solution: Checkpoint

Simple Version

Periodically:

(1) Do not accept new transactions;

(2) Wait until all transaction finish;

(3) Flush all log records to disk (log);

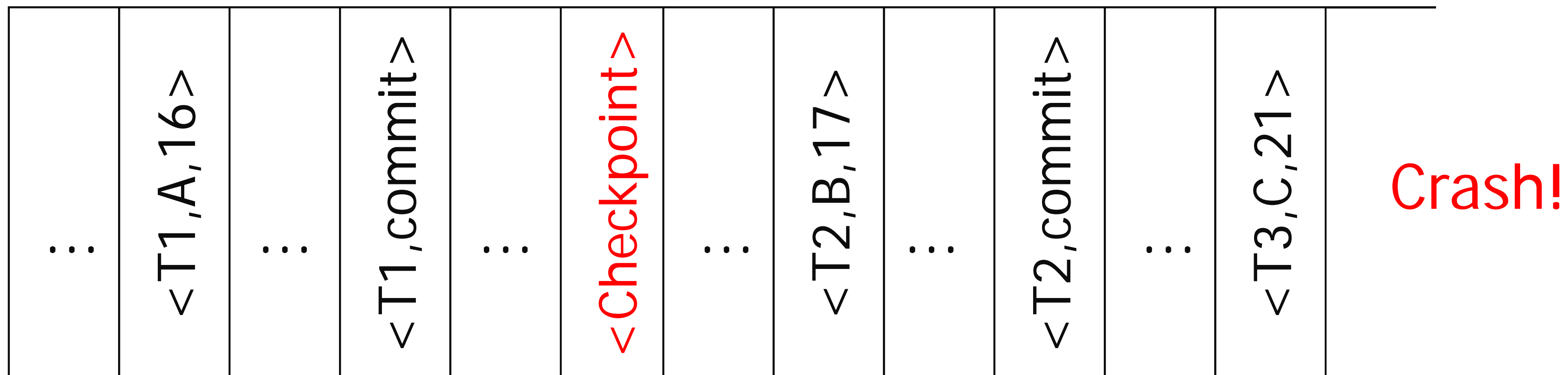
(4) Flush all buffers to disk (DB);

(5) Write “checkpoint” record on disk (log);

(6) Resume transaction proceeding

Example: what to do at recovery?

A Redo Log (disk):



Recovery does not need to go beyond checkpoint!

Better: Fuzzy Checkpointing

- Want to be able to checkpoint without bringing the system to a halt
- Especially if transactions run for a long time
- Solution: non-quiescent checkpointing
  - Will get to this soon…

Key Drawbacks

- Undo logging:
  - Must FORCE data to be on disk so as to commit
- Redo logging:
  - Need to keep all modified data in memory until commit
  - In other words, other actions cannot STEAL those space in buffer.

Logging Taxonomy

	No STEAL	STEAL
No FORCE	Redo Logging	Undo/Redo Logging
FORCE	No Logging	Undo Logging

Solution: Undo/Redo Logging

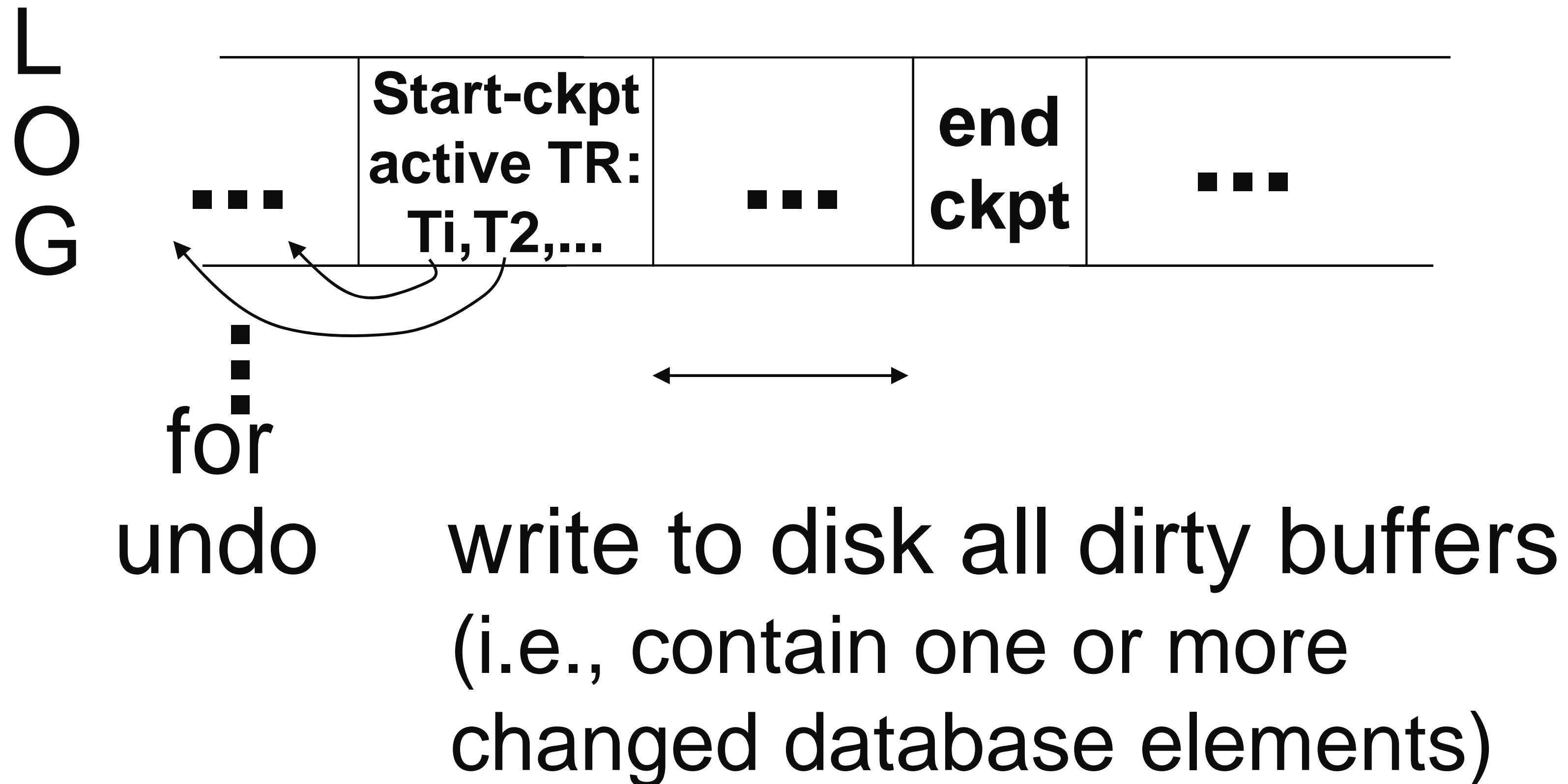
Undo/Redo Logging Rules

(1) Element X can be flushed before or after Ti commit

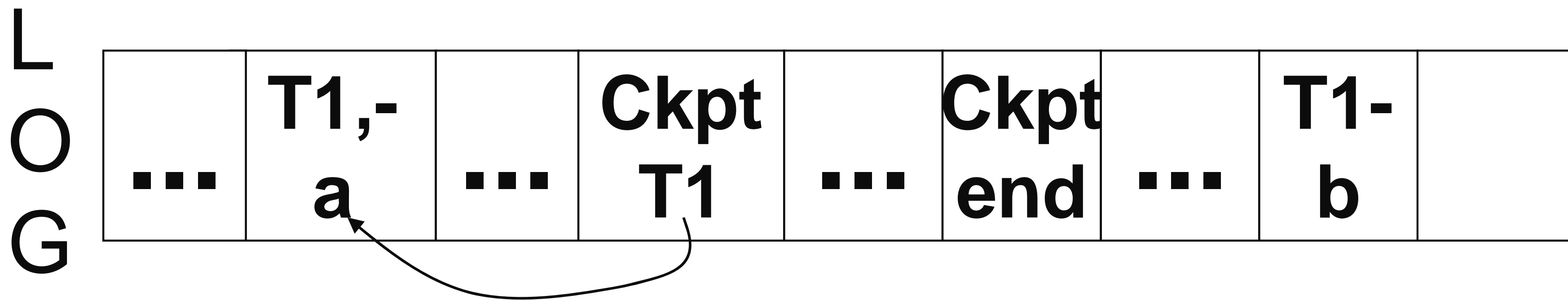
(2) Before modifying X on disk, all corresponding log records appear on disk

(3) Flush log before commit

Non-quiescent Checkpoint

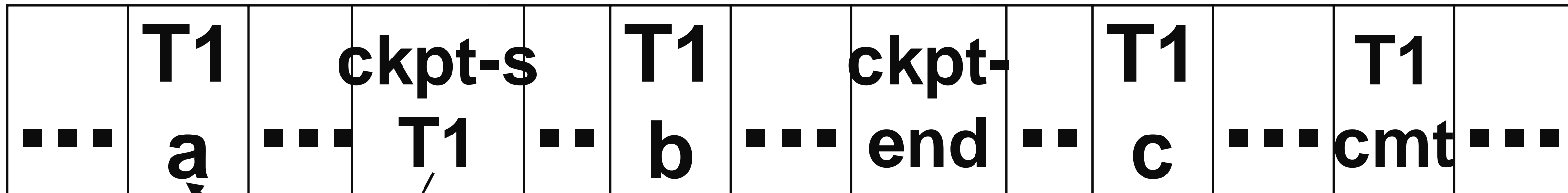


Examples



Undo T1 (undo a,b)

Examples



Redo T1 (redo b,c)

Recovery Process

- The undo/redo recovery policy
  - Redo committed transactions
  - Undo uncommitted transactions.
- **Backward pass** (end of log → latest checkpoint start)
  - Construct set C of transactions that committed since checkpoint start
  - Undo all actions of transactions not in C
- **Forward pass** (latest checkpoint start → end of log)
  - Redo all actions of transactions in C
- Alternatively, can also perform the redo before the undo.

Problems

- In either case, the following can happen.
  - Transaction T1 has committed and is redone.
  - However, T1 has read X written by transaction T2 which has not committed and is undone.
  - This situation needs to be avoided, since the resulting database state is inconsistent.
- Concurrency control ensures that this situation is avoided (to be discussed later).

Failure Models

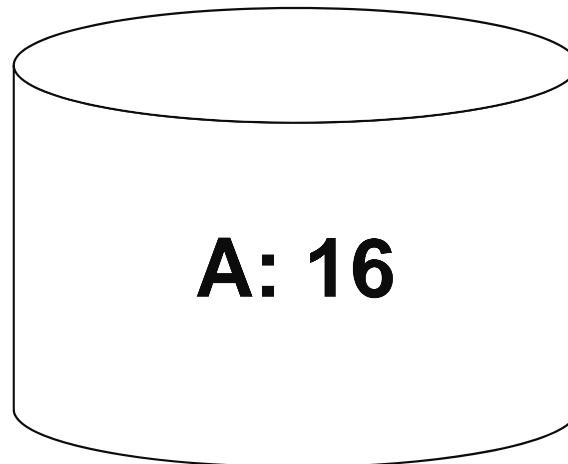
- Undesired expected:
  - System crash
    - Data on disk still there on restart
    - Solution: atomicity via logging
- Undesired unexpected:
  - Media failure
  - Catastrophic failure

  } Data on disks lost!

Media Failure

- Loss of non-violate storage

A
C
I
Durability



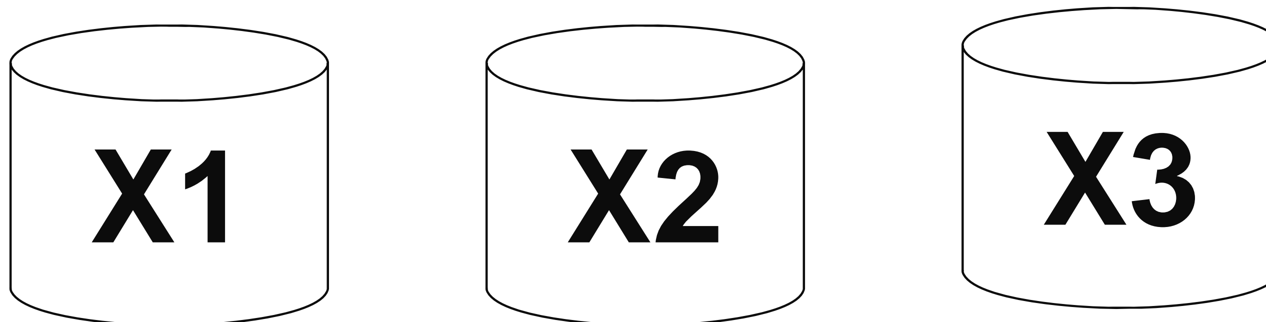
Solution: Make copies of data!

Example 1: Redundant Writes, Single Reads

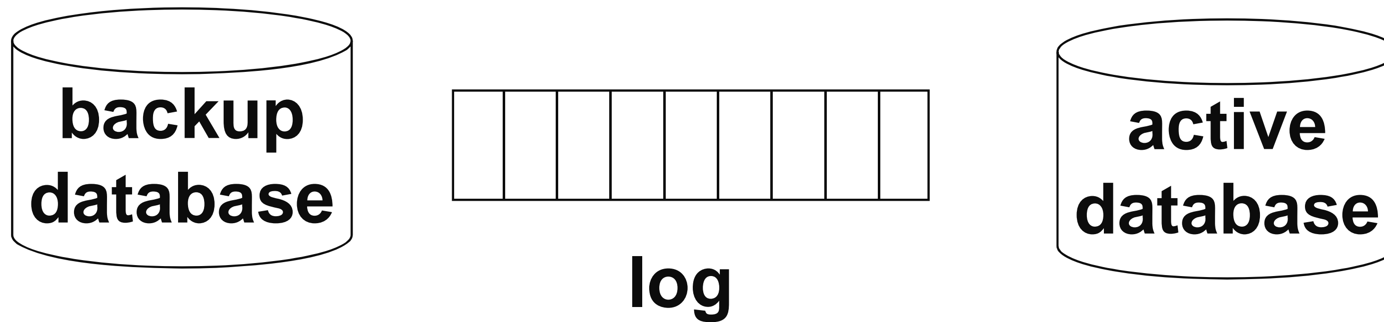
- Keep N copies on separate disks
- Output(X) → N outputs
- Input(X) → Input one copy
  - If okay, done
  - Else try another one
- Assumes bad data can be detected

Example 2: Triple Modular Redundancy

- Keep 3 copies on separate disks
- Output(X) → three outputs
- Input(X) → three inputs + votes



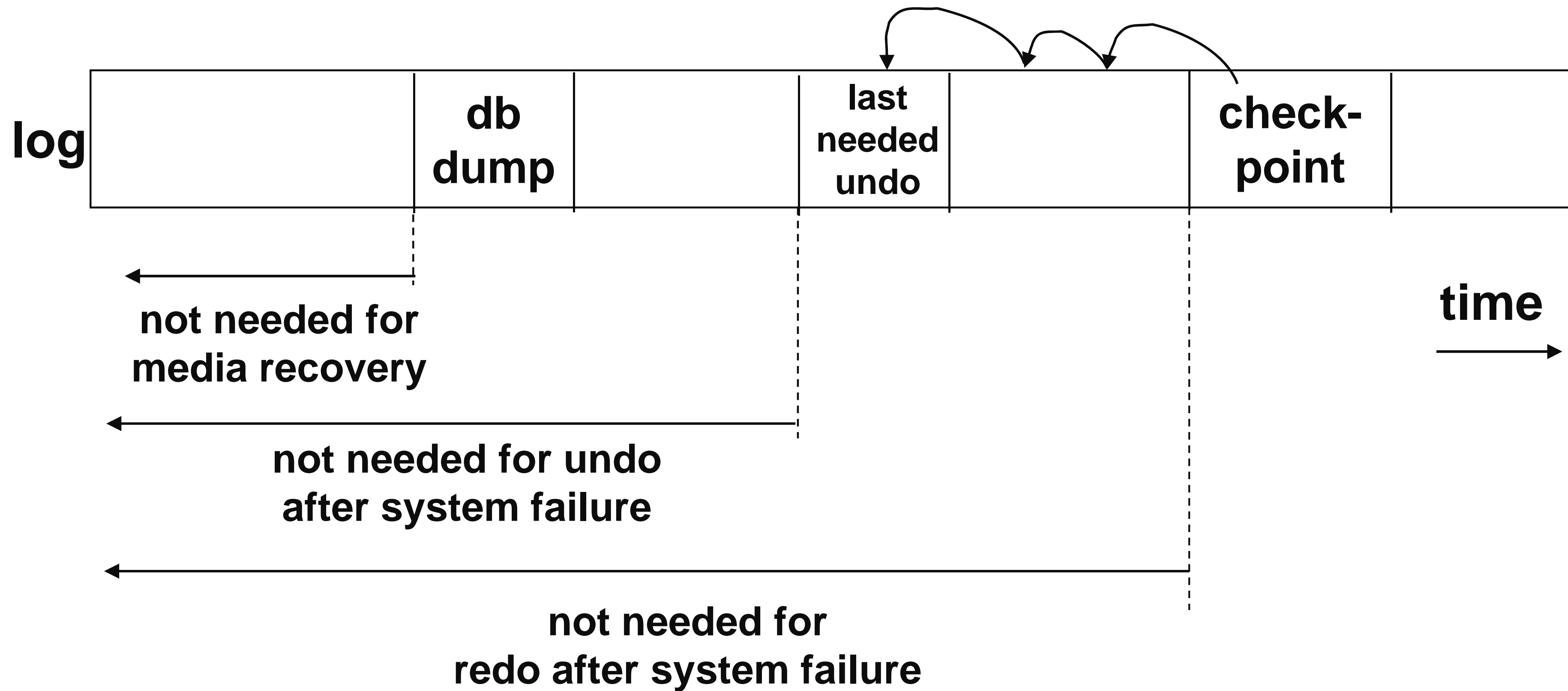
Example 3: DB Dump + Log



If active database is lost:
(1) Restore active database from backup
(2) Bring up-to-date using redo entries in log

Cannot use undo-only logging!

When Can Log Be Discarded?



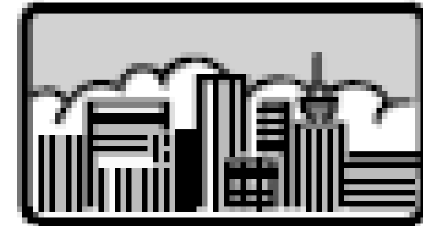
Catastrophic Failure

- Array of disks will not help in case of:
  - Fire
  - Explosion
  - Earthquake
  - Godzilla

  - Also, vandalism, viruses

Solution

- Geographically distributed copies!



Summary

- Consistency of data
- One source of problems: failures
  - Logging
  - Redundancy
- Another source of problems
  - Data sharing…. next