Transactions and Failure Recovery 2

Instructor: Matei Zaharia

CustomTable Showcase

Outline

Recap from last time

Redo logging

Undo/redo logging

External actions

Media failures

Outline

Recap from last time

Redo logging

Undo/redo logging

External actions

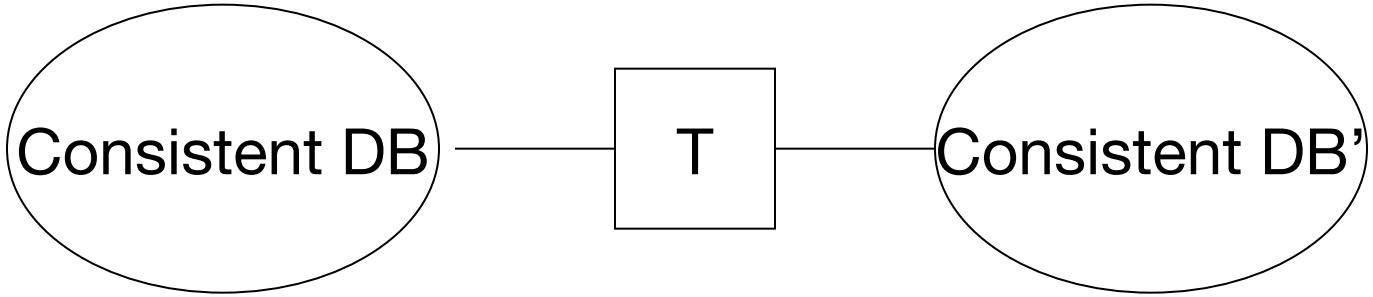
Media failures

Defining Correctness

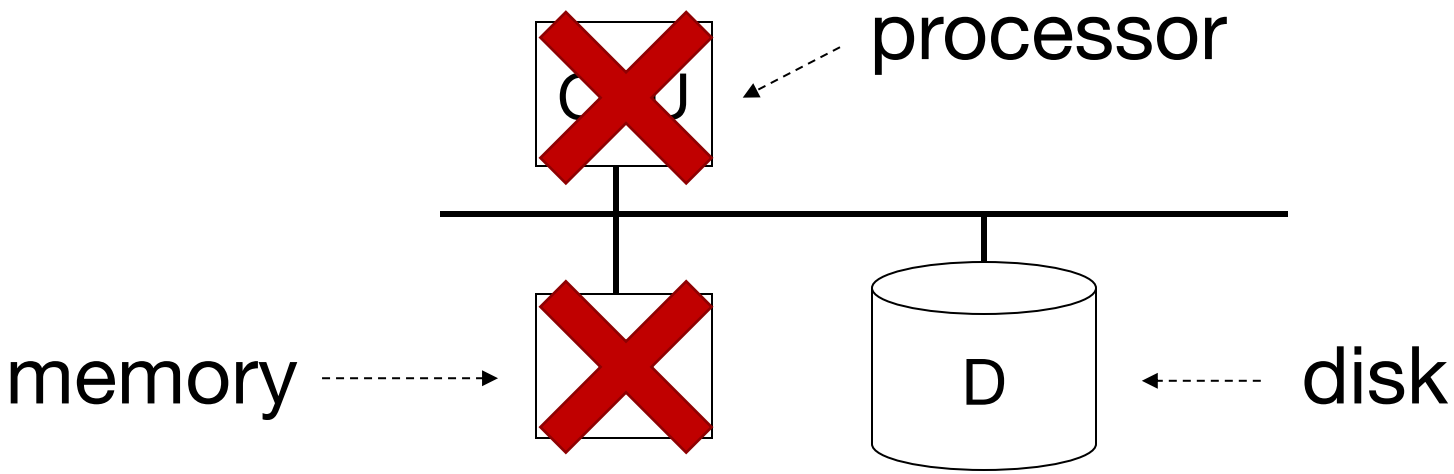
Constraint: Boolean predicate about DB state (both logical & physical data structures)

Consistent DB: satisfies all constraints

Transaction: Collection of Actions that Preserve Consistency



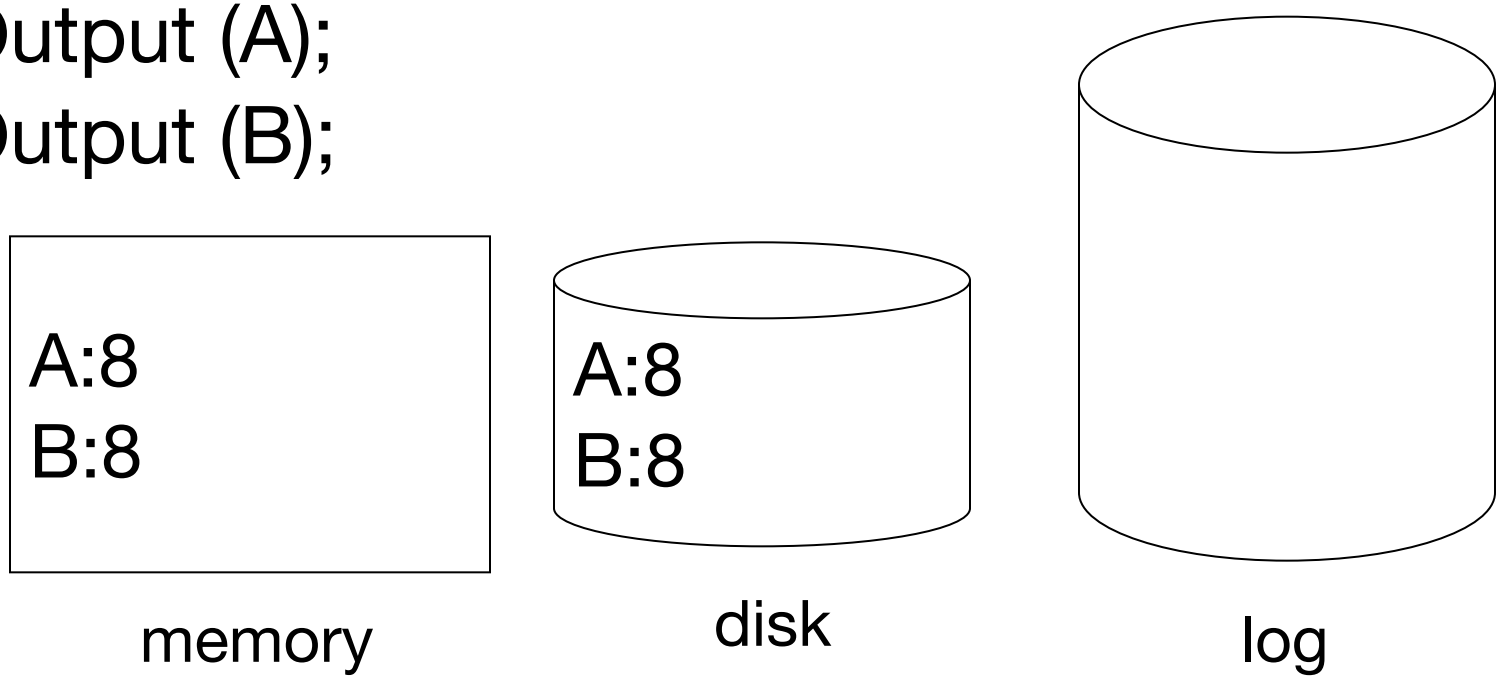
Our Failure Model



Fail-stop failures of CPU & memory, but disk survives

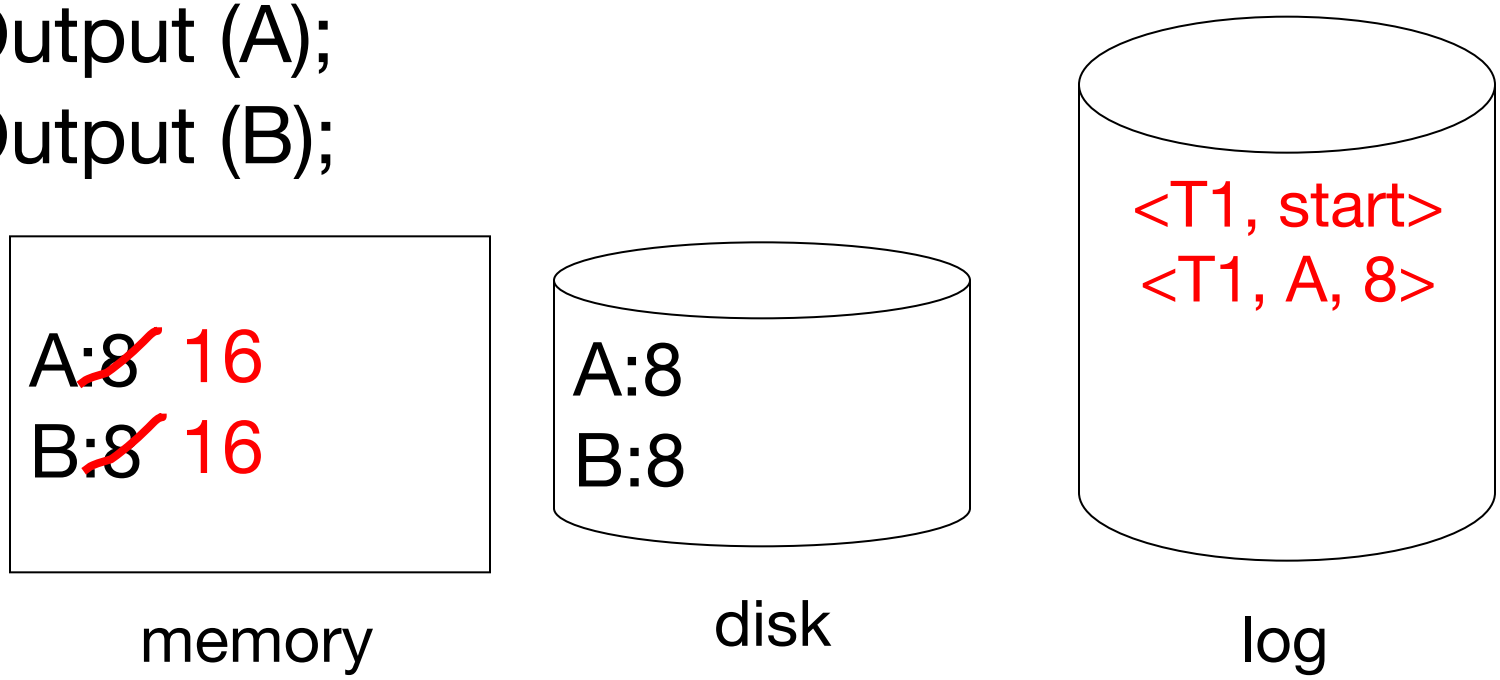
Undo Logging (Immediate modification)

T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



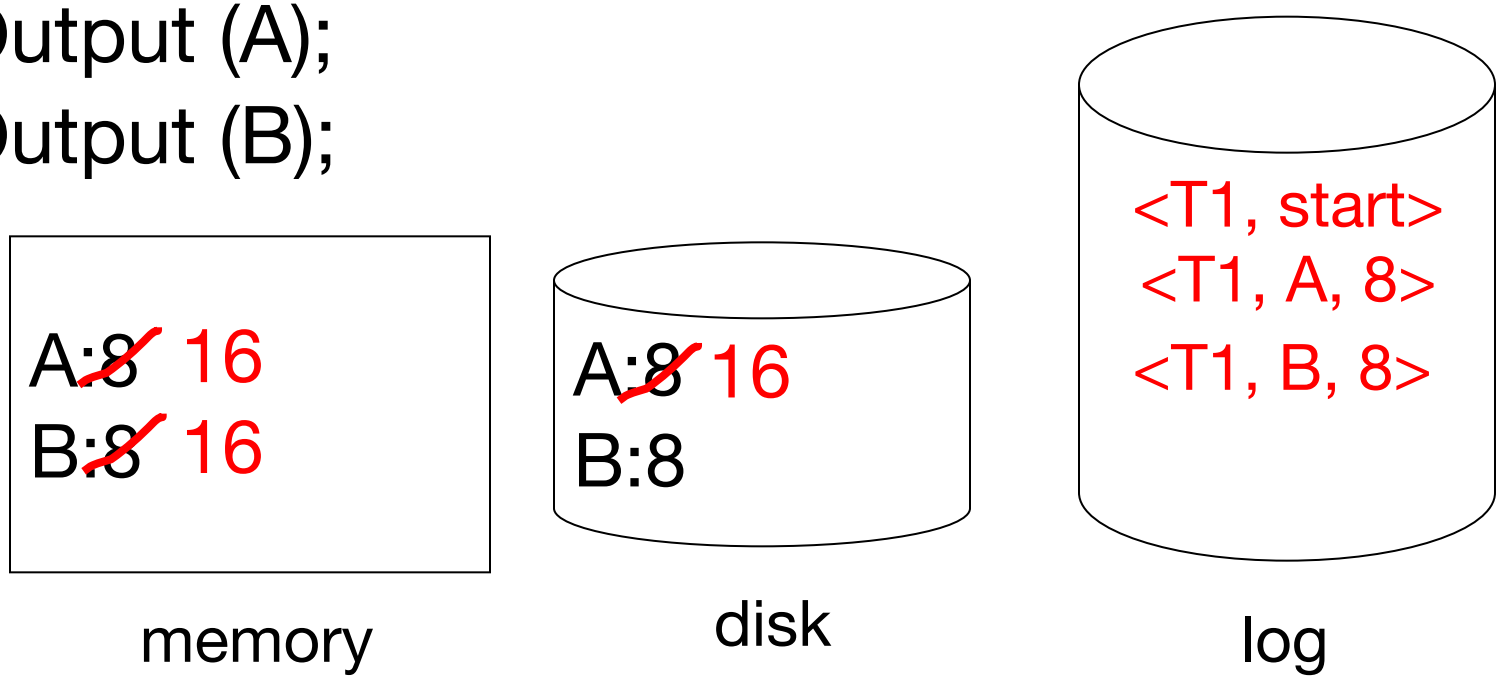
Undo Logging (Immediate modification)

T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



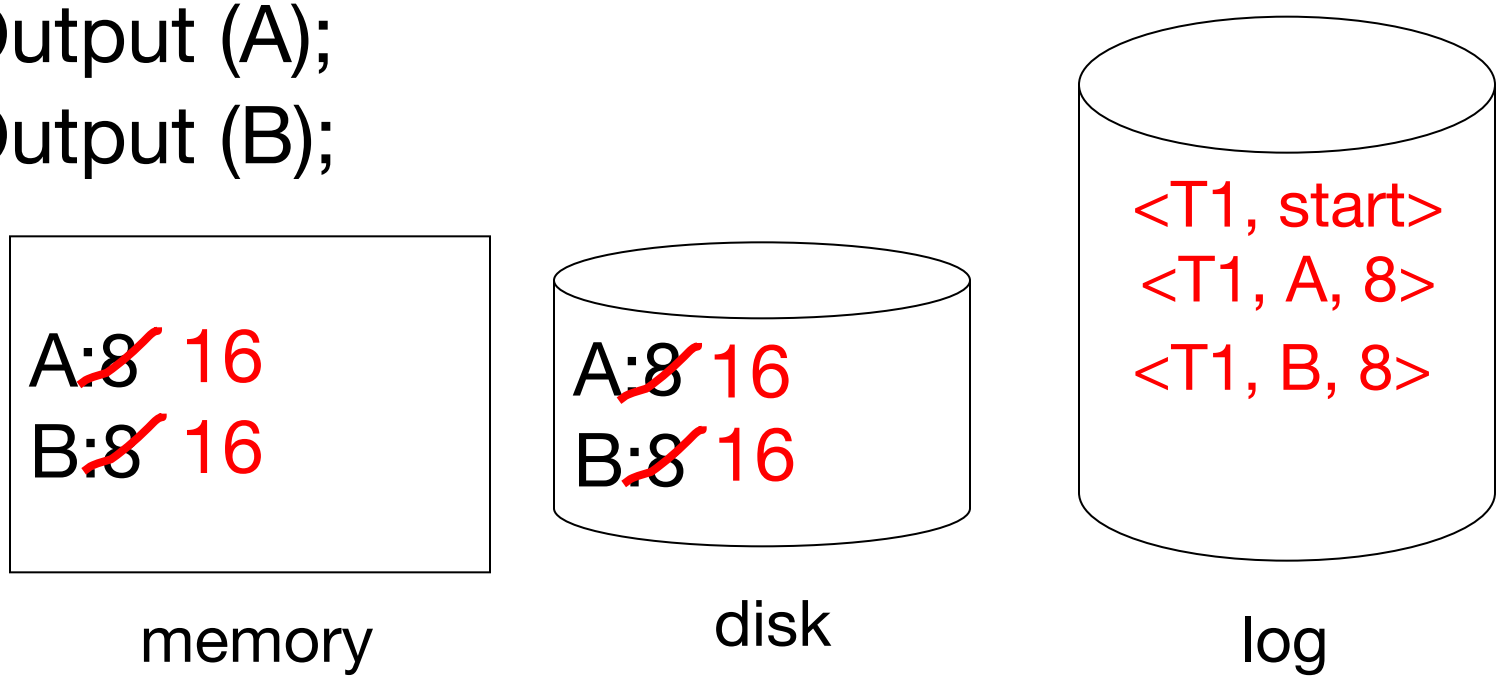
Undo Logging (Immediate modification)

T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



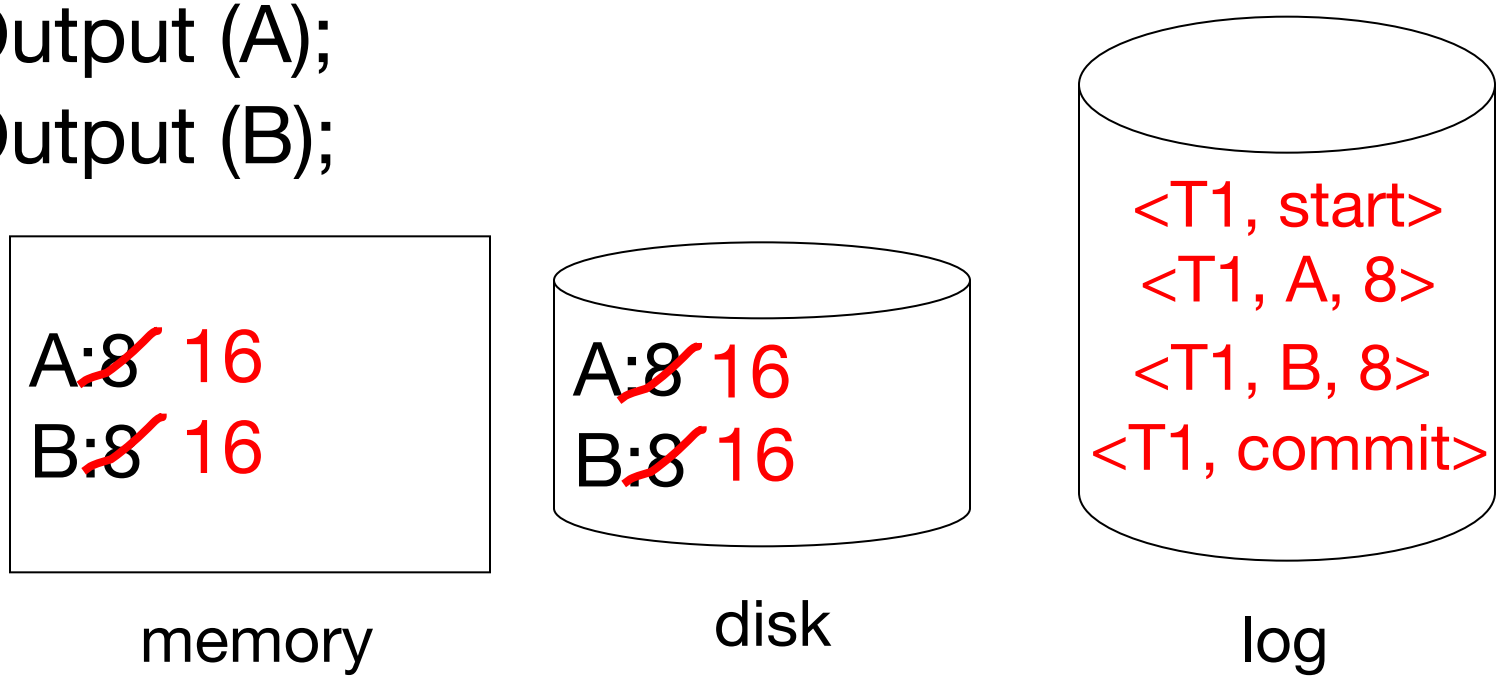
Undo Logging (Immediate modification)

T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);



Undo Logging (Immediate modification)

T1: Read (A,t); $t \leftarrow t \times 2$ A=B
Write (A,t);
Read (B,t); $t \leftarrow t \times 2$
Write (B,t);
Output (A);
Output (B);

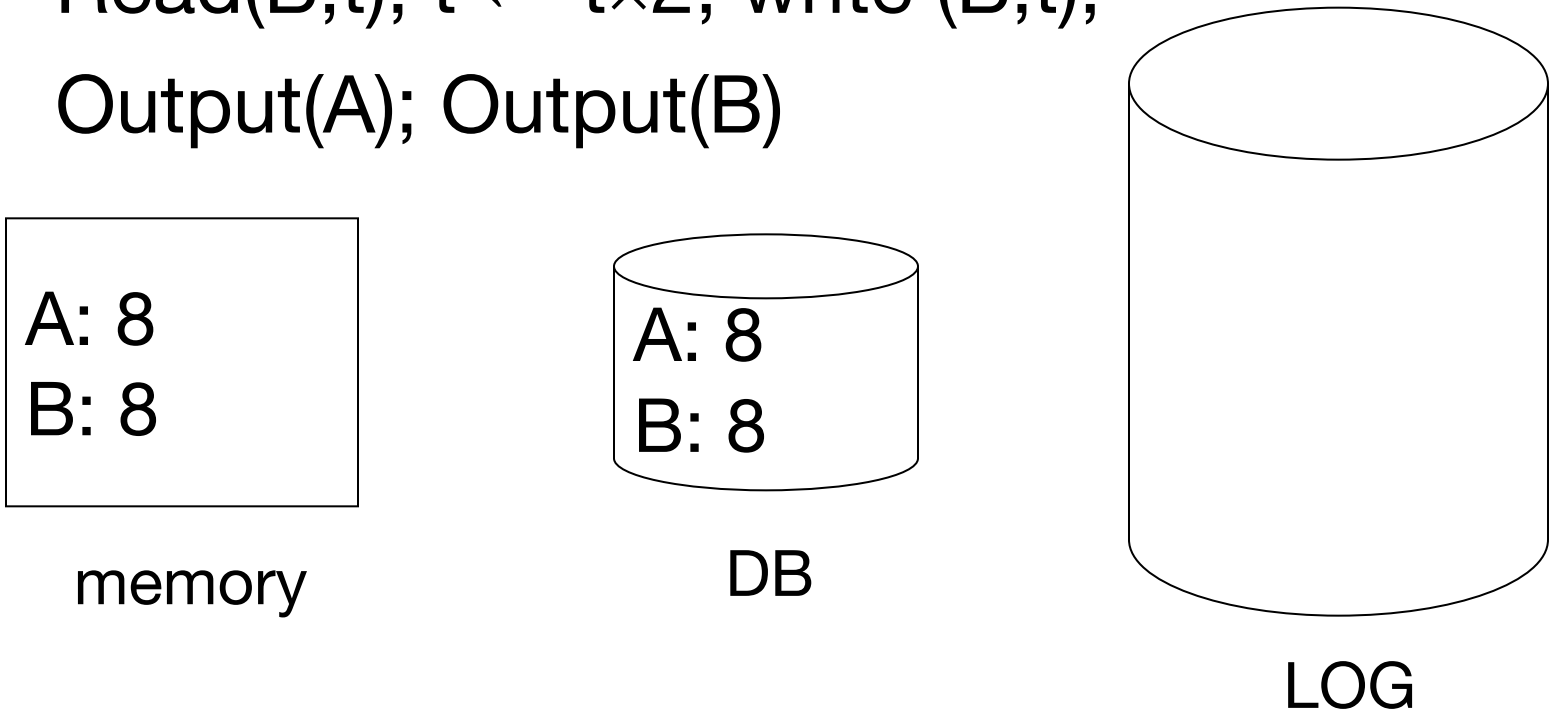


Redo Logging (deferred modification)

T1: Read(A,t); t ← t×2; write (A,t);

Read(B,t); t ← t×2; write (B,t);

Output(A); Output(B)

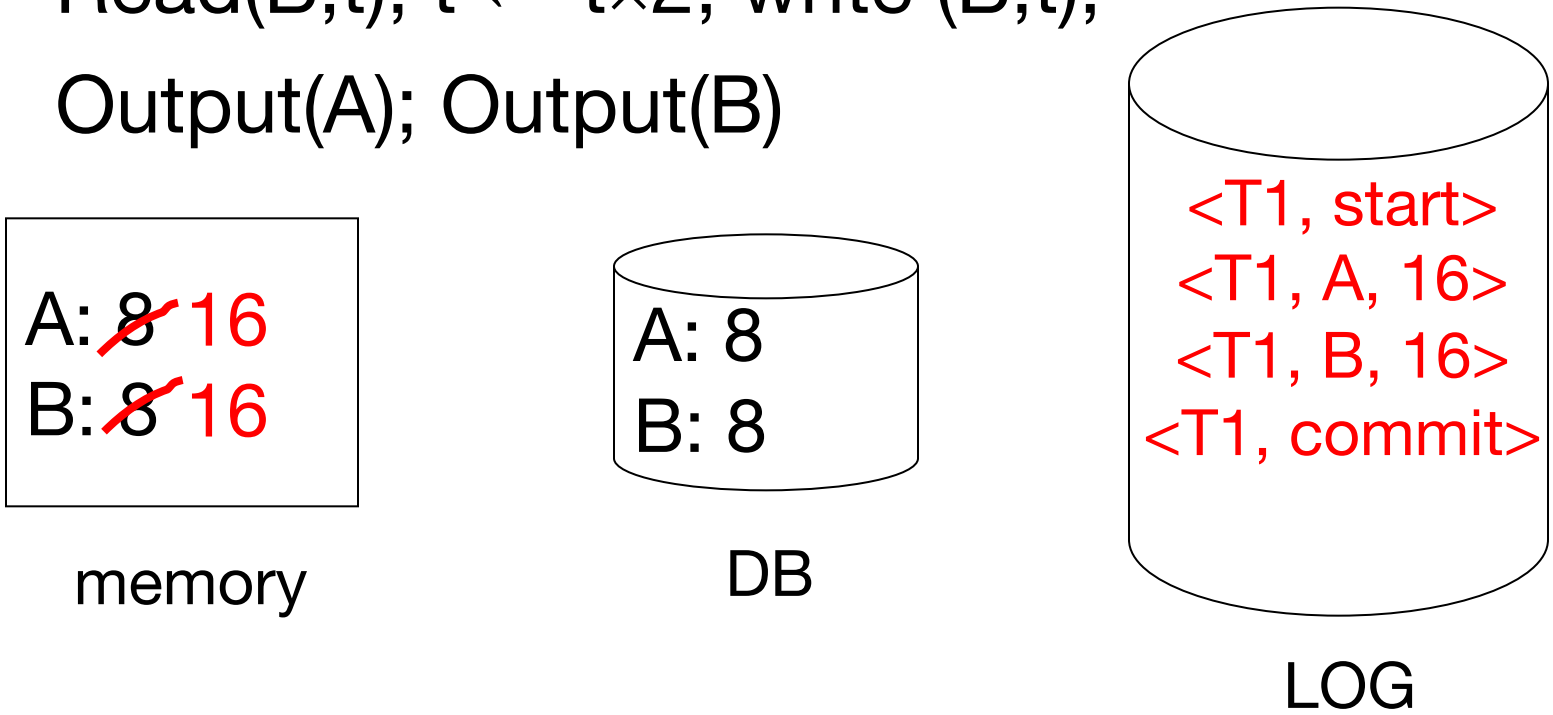


Redo Logging (deferred modification)

T1: Read(A,t); t ← t×2; write (A,t);

Read(B,t); t ← t×2; write (B,t);

Output(A); Output(B)

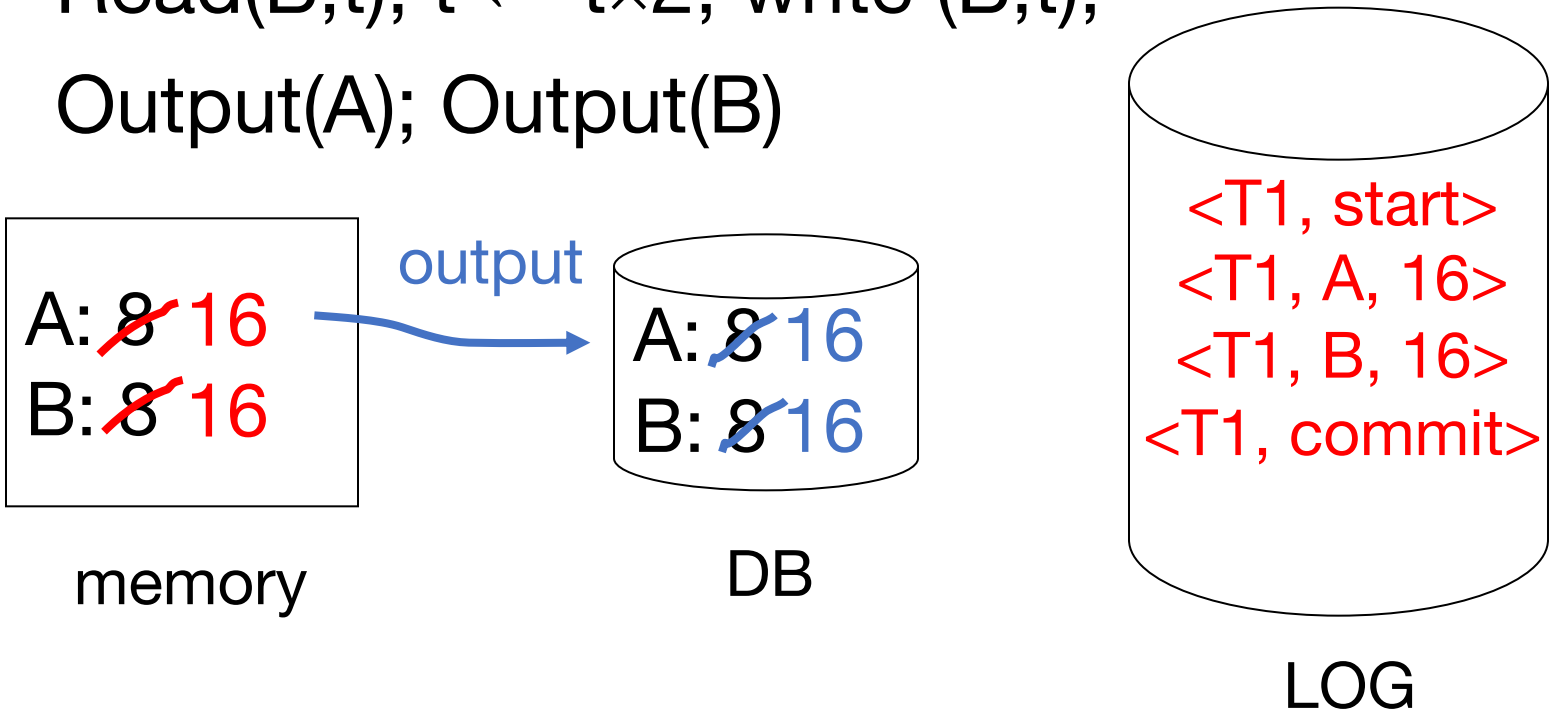


Redo Logging (deferred modification)

T1: Read(A,t); t ← t×2; write (A,t);

Read(B,t); t ← t×2; write (B,t);

Output(A); Output(B)

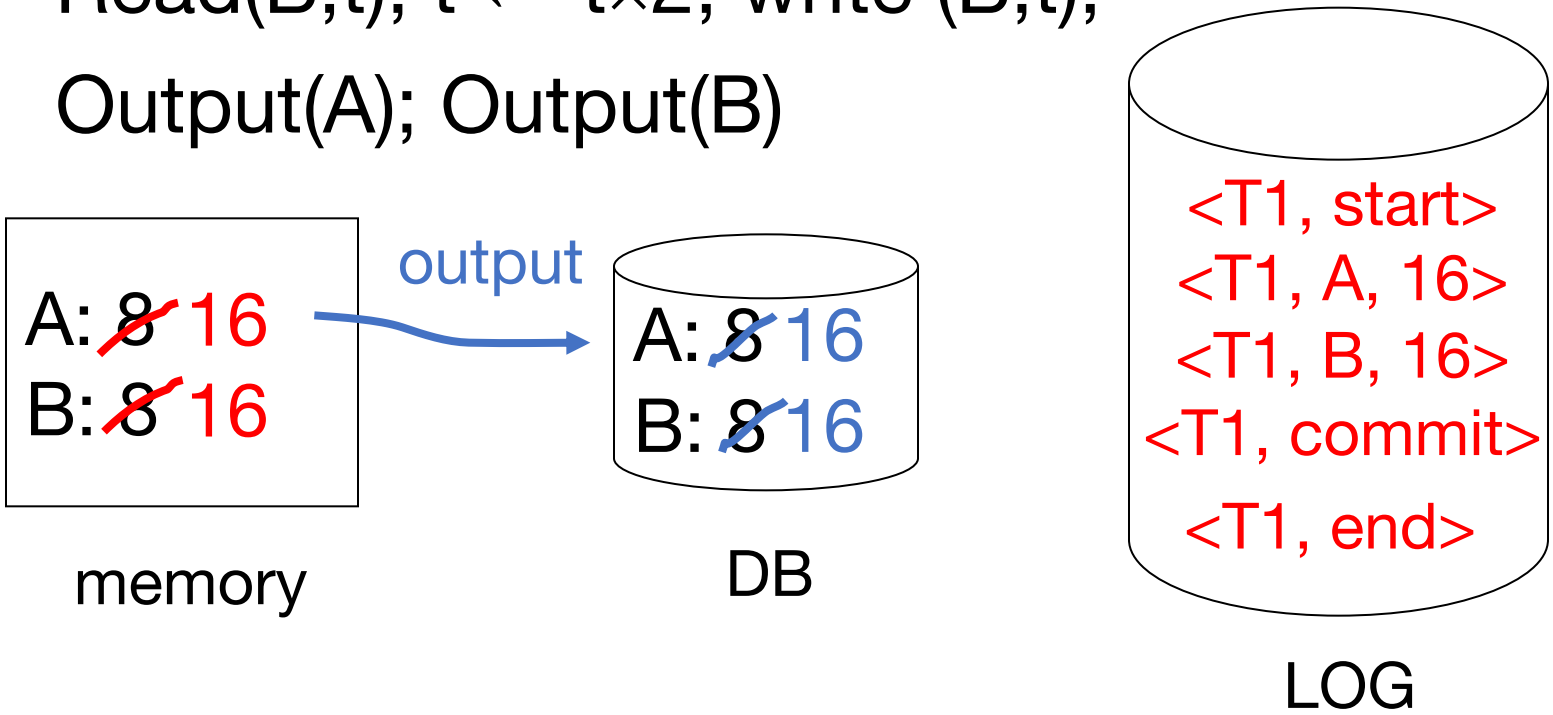


Redo Logging (deferred modification)

T1: Read(A,t); t ← t×2; write (A,t);

Read(B,t); t ← t×2; write (B,t);

Output(A); Output(B)



Redo Logging Rules

1. For every action, generate redo log record (containing new value)
2. Before X is modified on disk (in DB), all log records for transaction that modified X (including commit) must be on disk
3. Flush log at commit
4. Write END record after DB updates are flushed to disk

Recovery Rules: Redo Logging

(1) Let S = set of transactions with
<Ti, commit> and no <Ti, end> in log

(2) For each <Ti, X, v> in log, in forward order
(earliest $\rightarrow$ latest) do:

- if Ti $\in$ S then $\begin{cases} \text{Write}(X, v) \\ \text{Output}(X) \end{cases}$

(3) For each Ti $\in$ S, write <Ti, end>

Combining <Ti, end> Records

Want to delay DB flushes for hot objects

Say X is branch balance:
T1: ... update X...
T2: ... update X...
T3: ... update X...
T4: ... update X...

<u>Actions:</u>
write X
output X
write X
output X
write X
output X
write X
output X

Combining <Ti, end> Records

Want to delay DB flushes for hot objects

Say X is branch balance:
T1: ... update X...
T2: ... update X...
T3: ... update X...
T4: ... update X...

<u>Actions:</u>
write X
~~output X~~
write X
~~output X~~
write X
~~output X~~
write X
output X

combined <end> record (checkpoint)

Solution: Checkpoints

Simple, naïve checkpoint algorithm:

1. Stop accepting new transactions
2. Wait until all transactions finish
3. Flush all log records to disk (log)
4. Flush all buffers to disk (DB) (do not discard buffers)
5. Write “checkpoint” record on disk (log)
6. Resume transaction processing

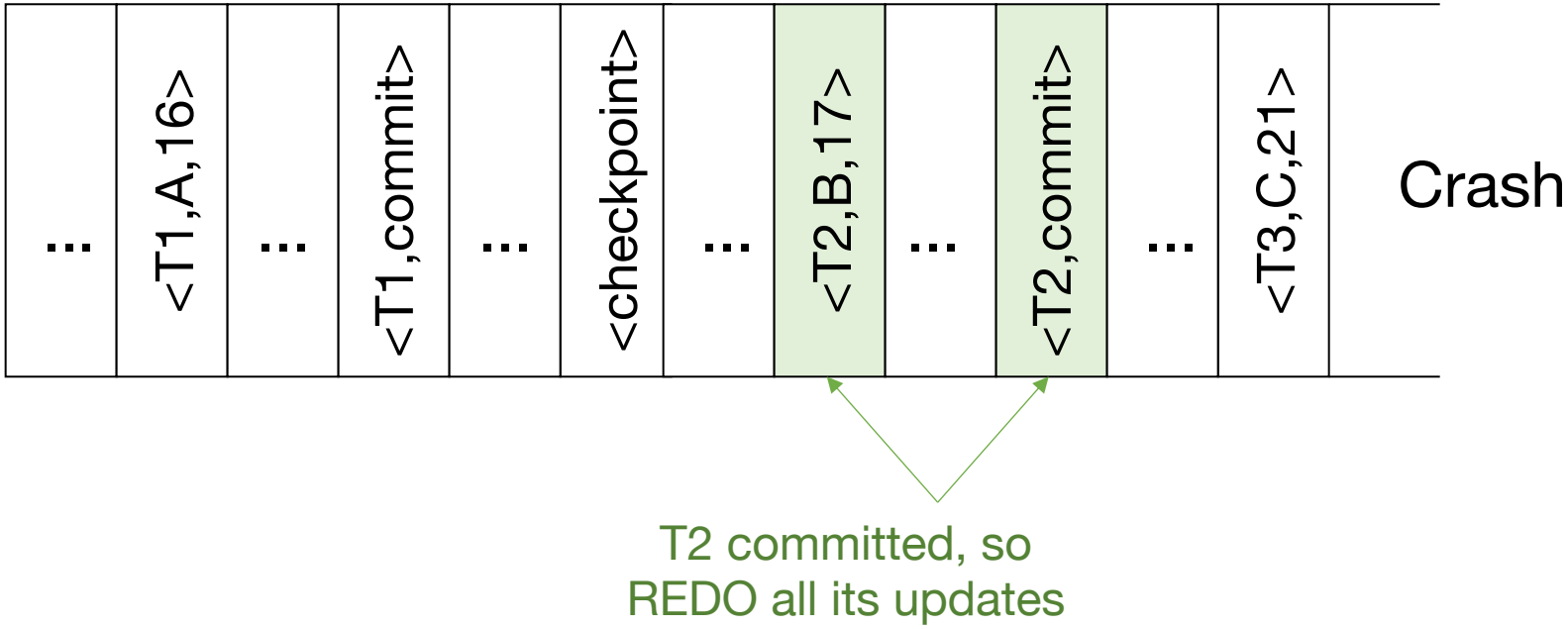
Redo Logging: What To Do at Recovery?

Redo log (disk):

...	<T1,A,16>	...	<T1,commit>	...	<checkpoint>	...	<T2,B,17>	...	<T2,commit>	...	<T3,C,21>	Crash

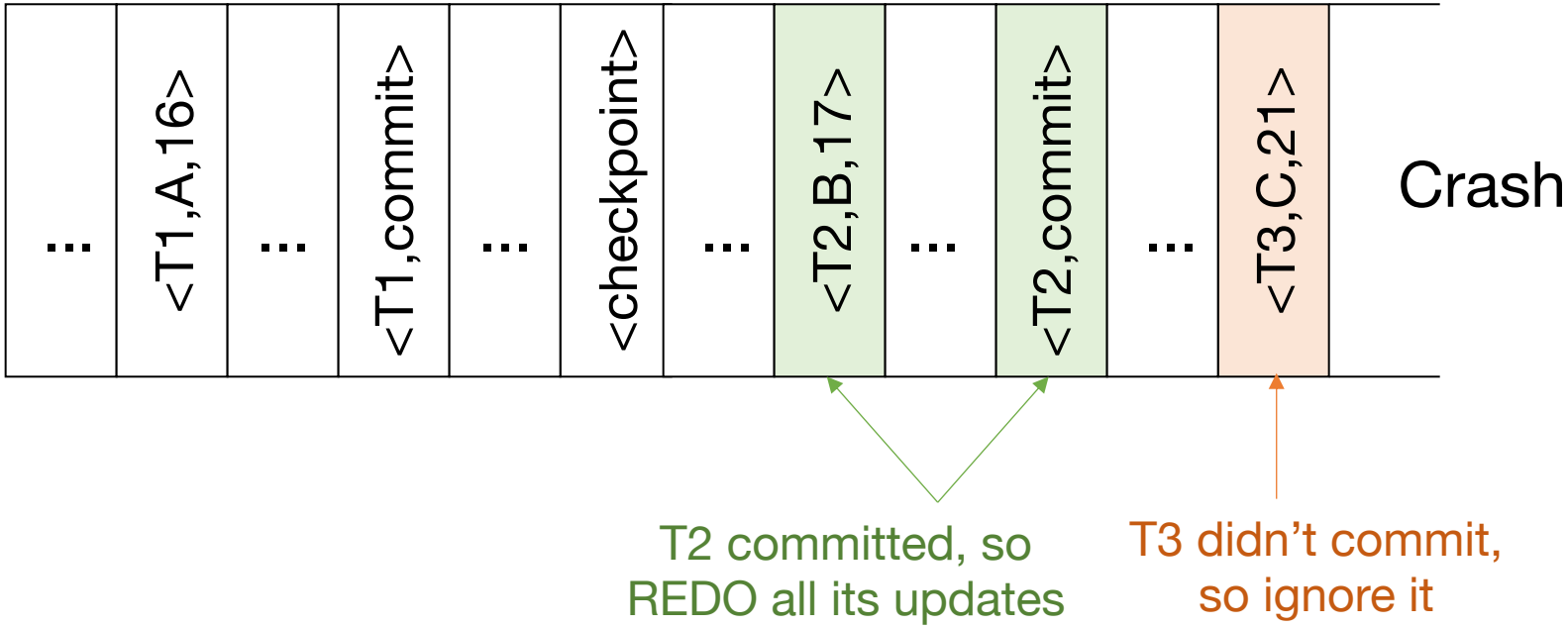
Redo Logging: What To Do at Recovery?

Redo log (disk):



Redo Logging: What To Do at Recovery?

Redo log (disk):



Problems with Ideas So Far

Undo logging: need to wait for lots of I/O to commit; can’t easily have backup copies of DB

Redo logging: need to keep all modified blocks in memory until commit

+

=

Solution: Undo/Redo Logging!

Update = <Ti, X, new X val, old X val>

(X is the object updated)

Undo/Redo Logging Rules

Object X can be flushed **before or after** Ti commits

Log record (with undo/redo info) must be flushed before corresponding data (WAL)

Flush log up to commit record at Ti commit

Undo/Redo Logging: What to Do at Recovery?

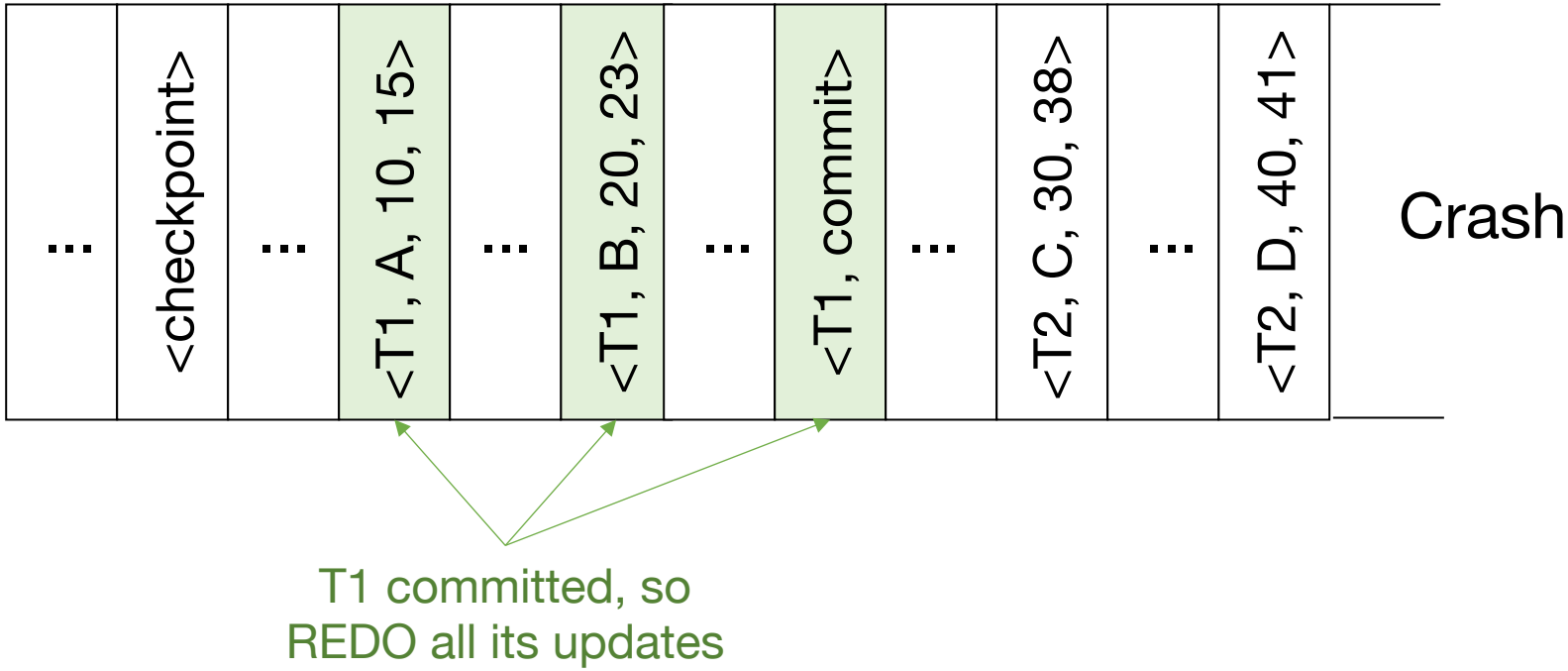
Undo/redo log (disk):

...
<checkpoint>
...
<T1, A, 10, 15>
...
<T1, B, 20, 23>
...
<T1, commit>
...
<T2, C, 30, 38>
...
<T2, D, 40, 41>

Crash

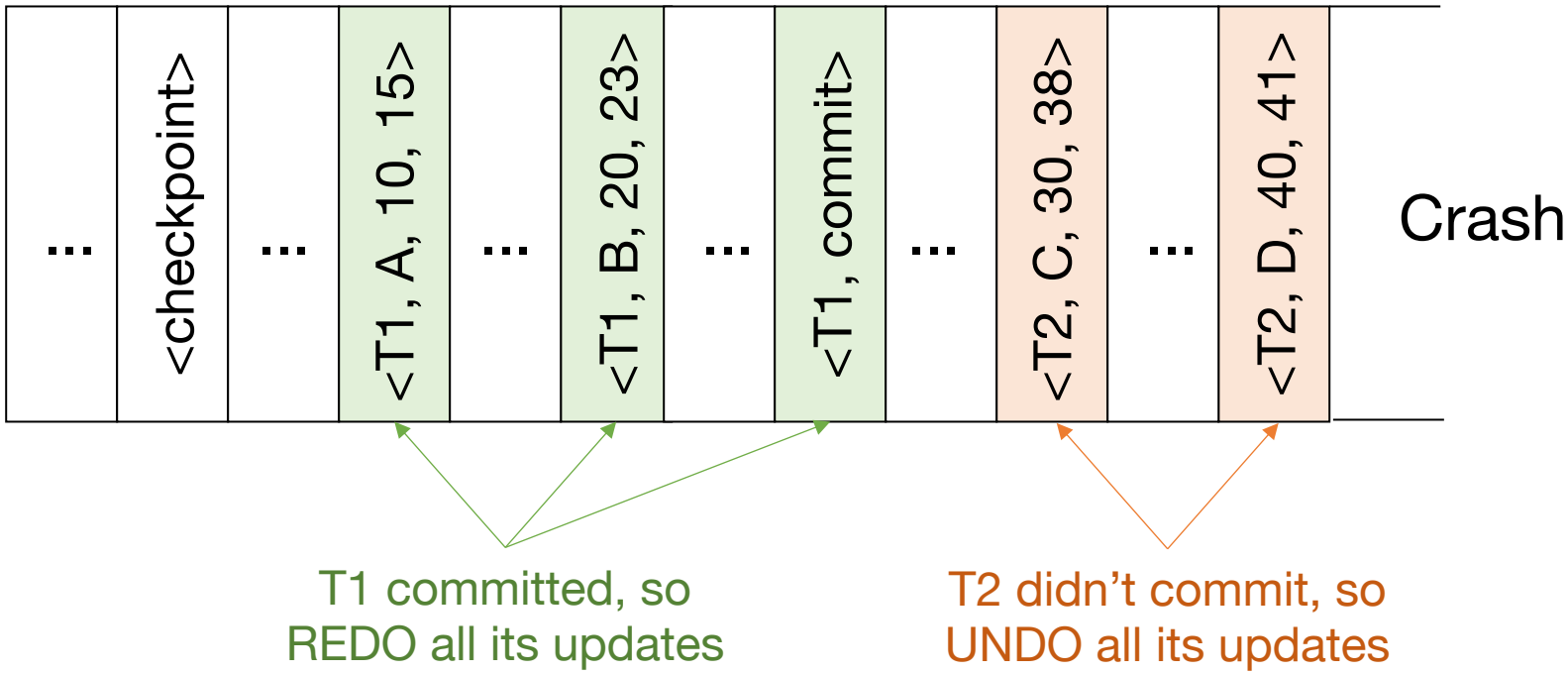
Undo/Redo Logging: What to Do at Recovery?

Undo/redo log (disk):

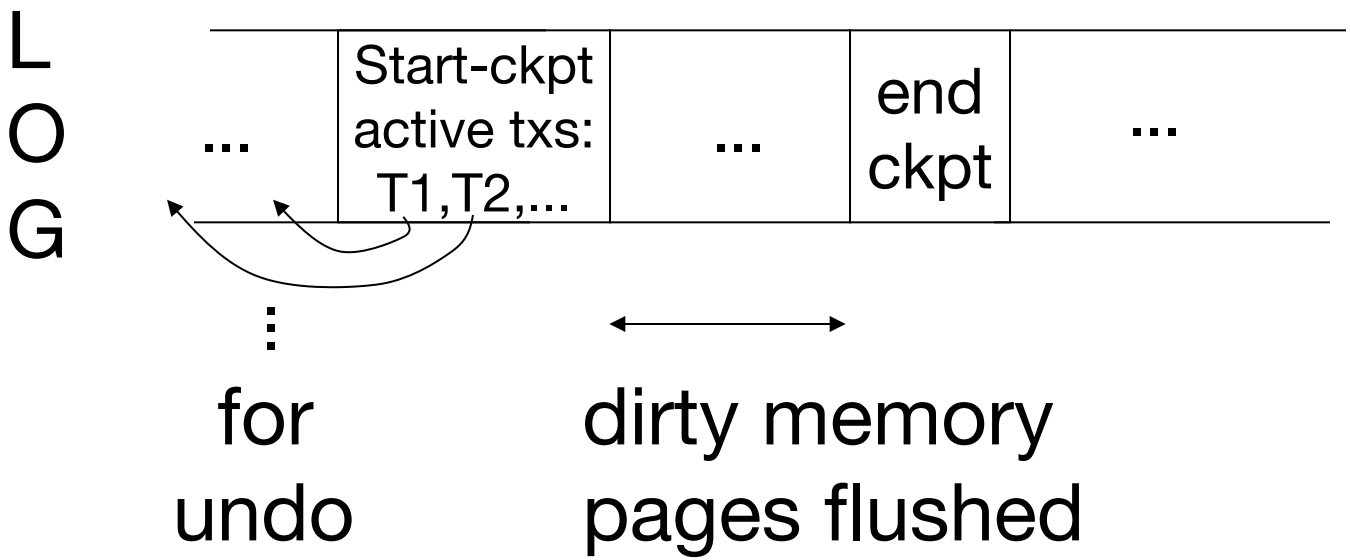


Undo/Redo Logging: What to Do at Recovery?

Undo/redo log (disk):



Non-Quiescent Checkpoints

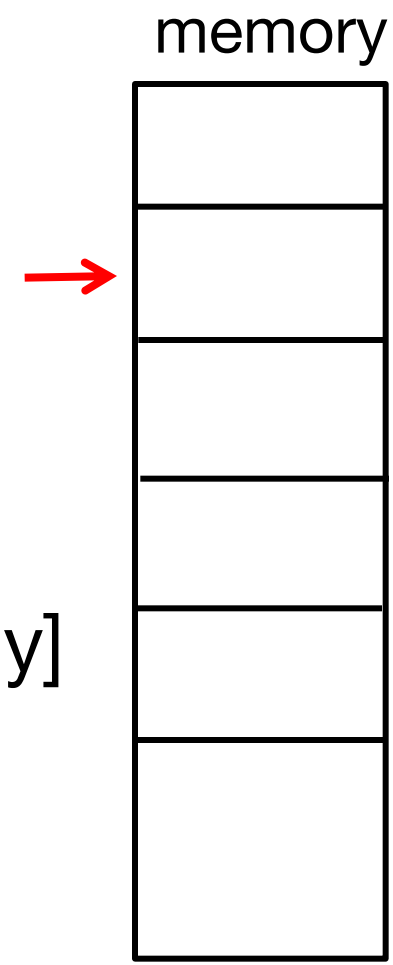


Non-Quiescent Checkpoints

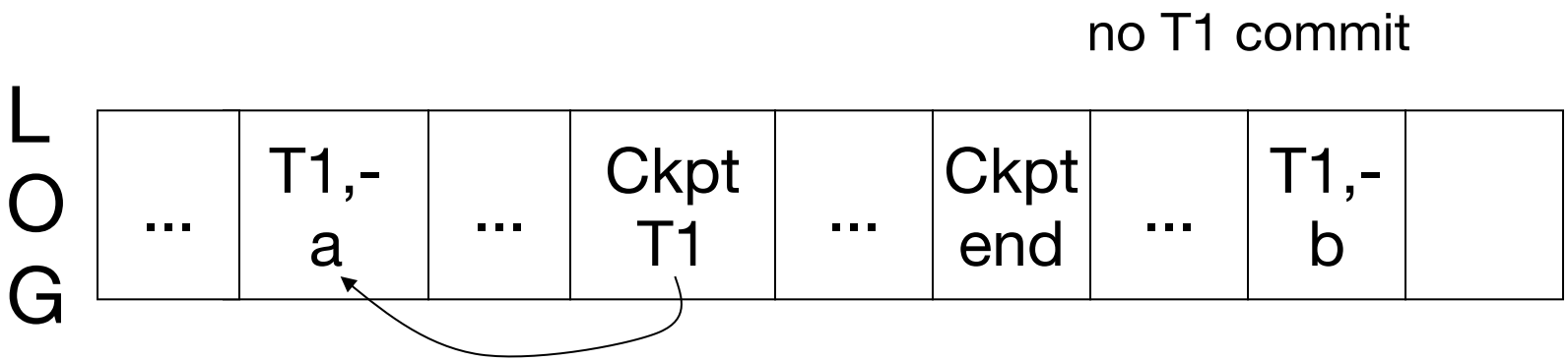
checkpoint process:

for i := 1 to M do
    Output(buffer i)

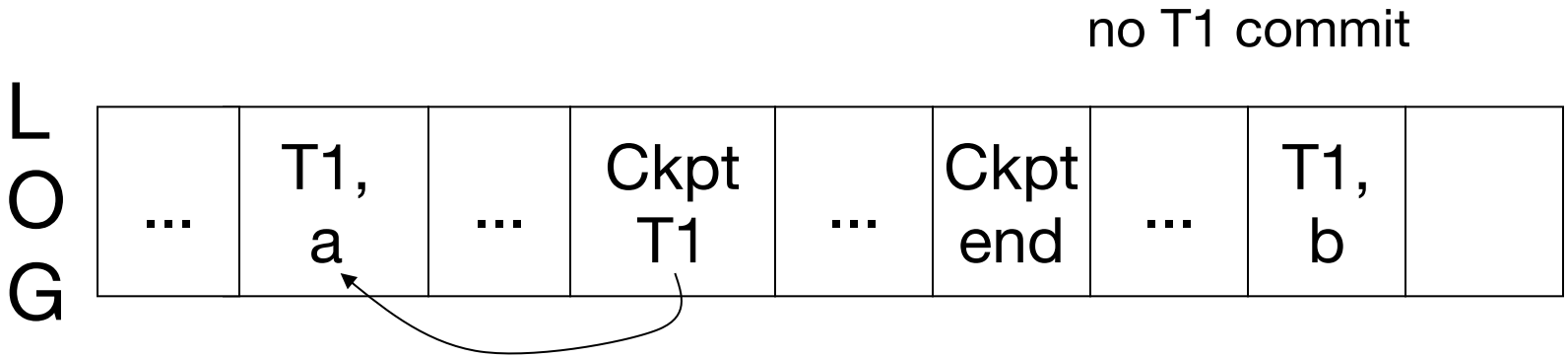
[transactions run concurrently]



Example 1: How to Recover?

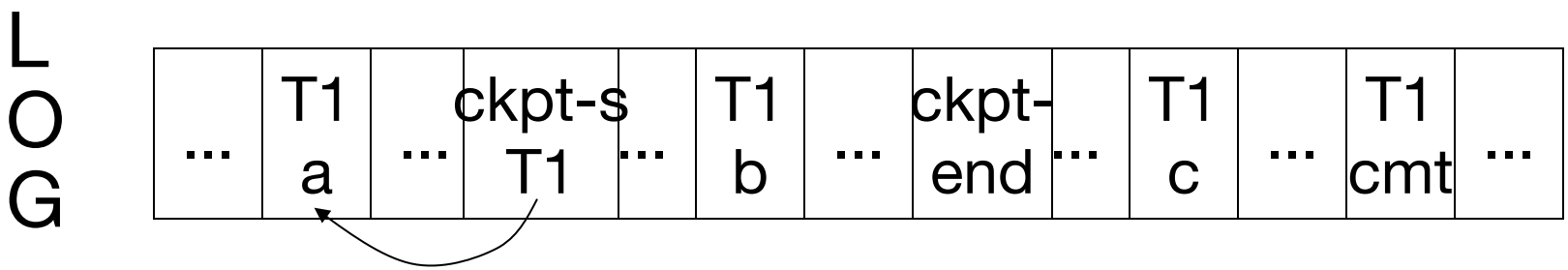


Example 1: How to Recover?

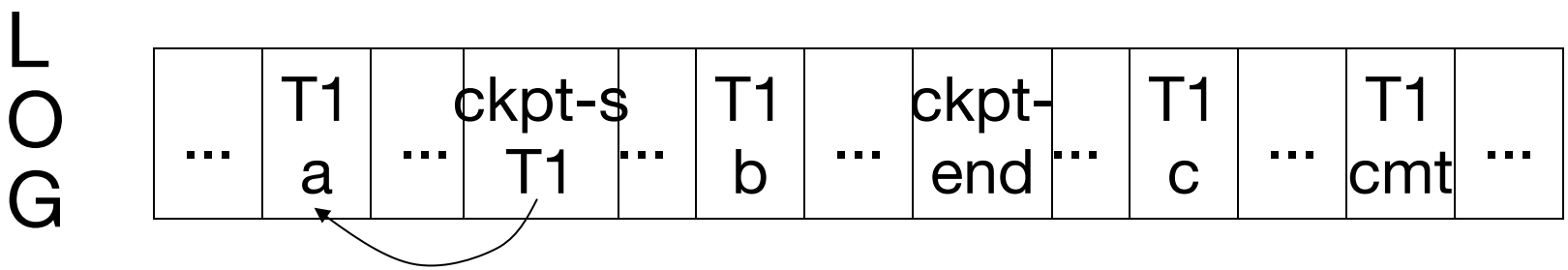


Undo T1 (undo a,b)

Example 2: How to Recover?

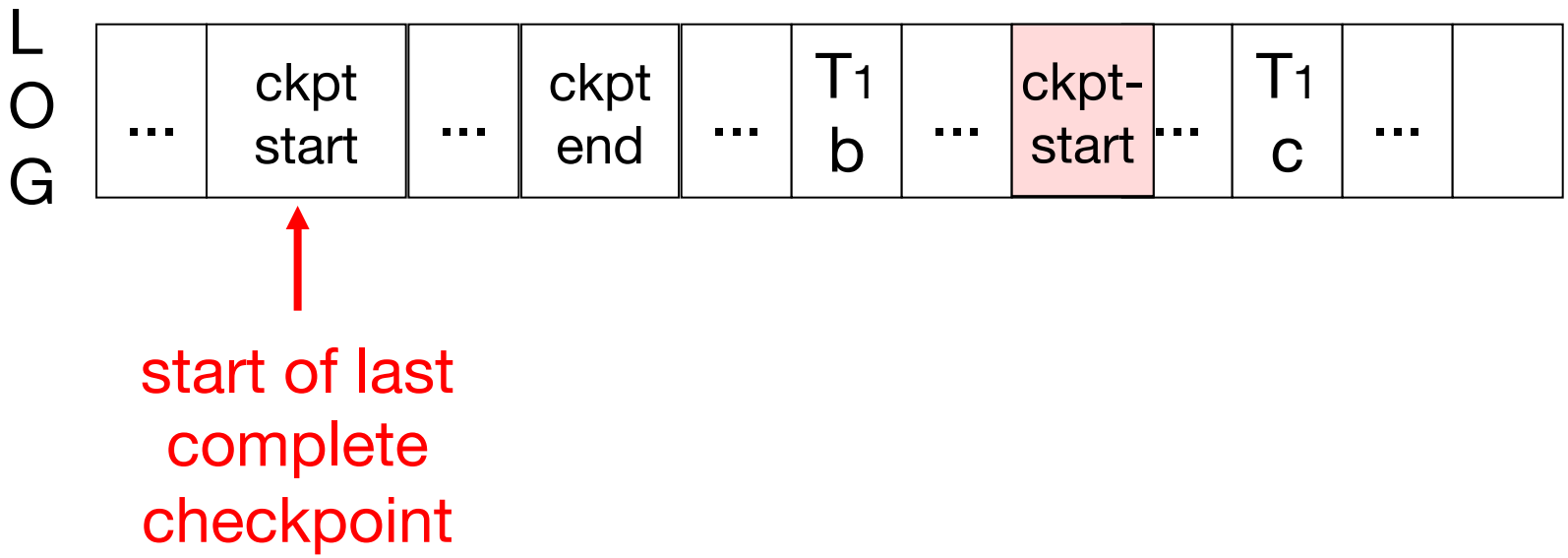


Example 2: How to Recover?



Redo T1 (redo b,c)

What if a Checkpoint Did Not Complete?



Start recovery from last complete checkpoint

Undo/Redo Recovery Algorithm

Backward pass (end of log → latest valid checkpoint start)

» construct set S of committed transactions

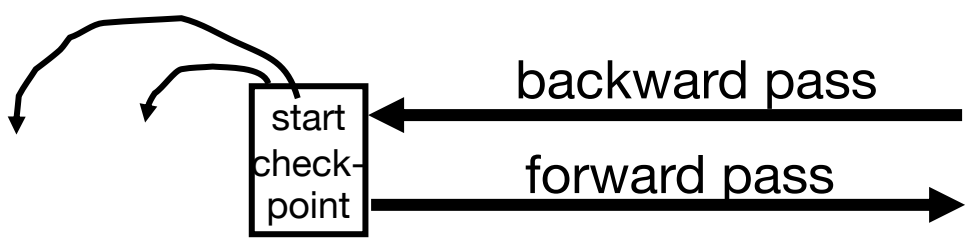
» undo actions of transactions not in S

Undo pending transactions

» follow undo chains for transactions in (checkpoint's active list) - S

Forward pass (latest checkpoint start → end of log)

» redo actions of all transactions in S



Outline

Recap from last time

Redo logging

Undo/redo logging

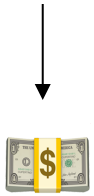
External actions

Media failures

External Actions

E.g., dispense cash at ATM

$$Ti = a_1\ a_2\ \ldots\ldots\ a_j\ \ldots\ldots\ a_n$$



Solution

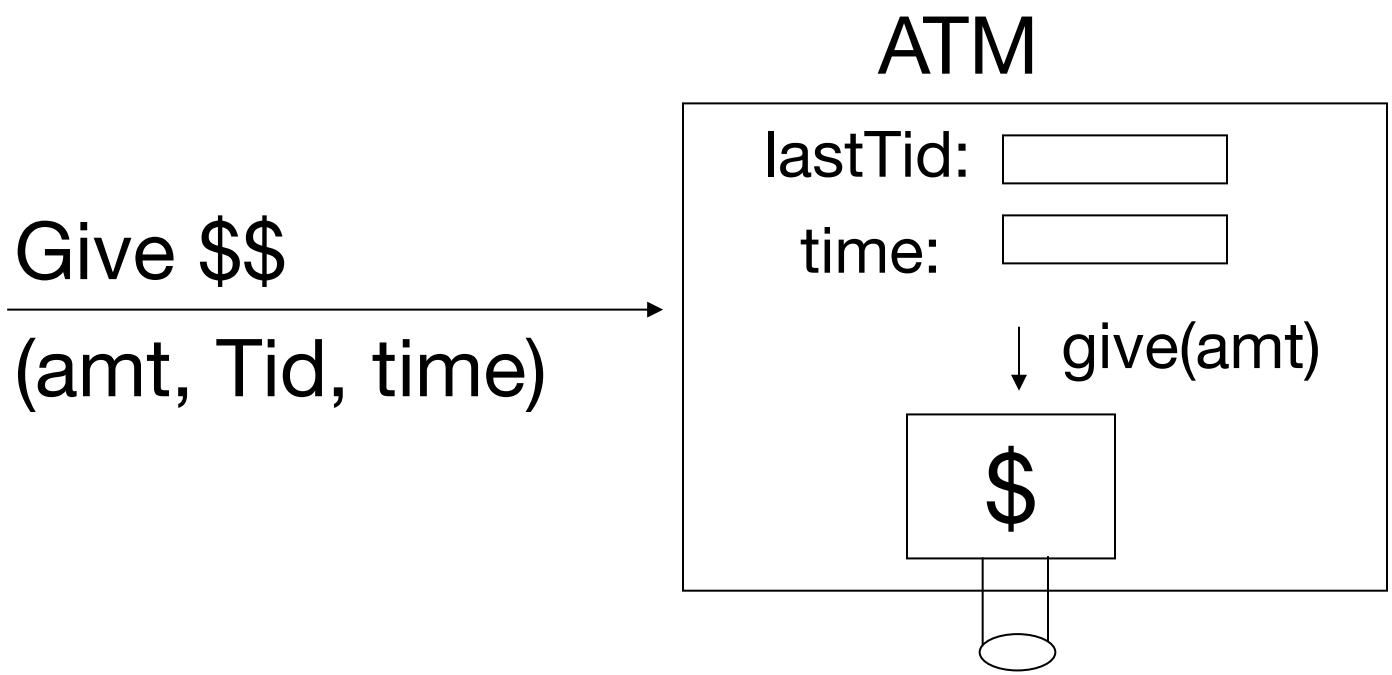
(1) Execute real-world actions after commit

(2) Try to make idempotent

Solution

(1) Execute real-world actions after commit

(2) Try to make idempotent



How Would You Handle These Other External Actions?

Charge a customer’s credit card

Cancel someone’s hotel room

Send data into a streaming system

Outline

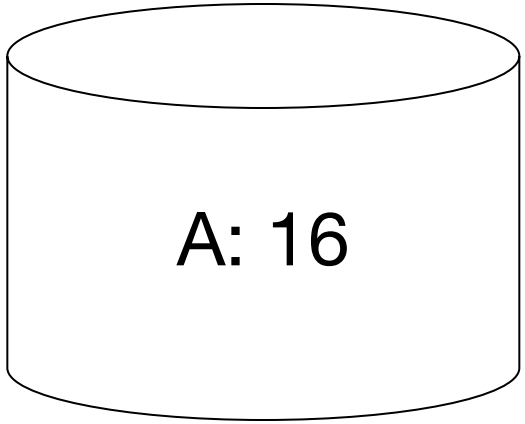
Recap from last time

Undo/redo logging

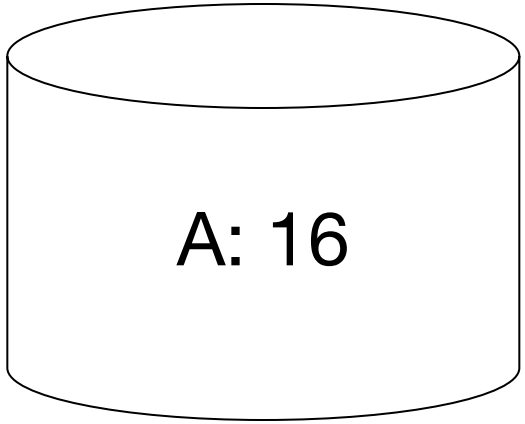
External actions

Media failures

Media Failure
(Loss of Nonvolatile Storage)



Media Failure
(Loss of Nonvolatile Storage)



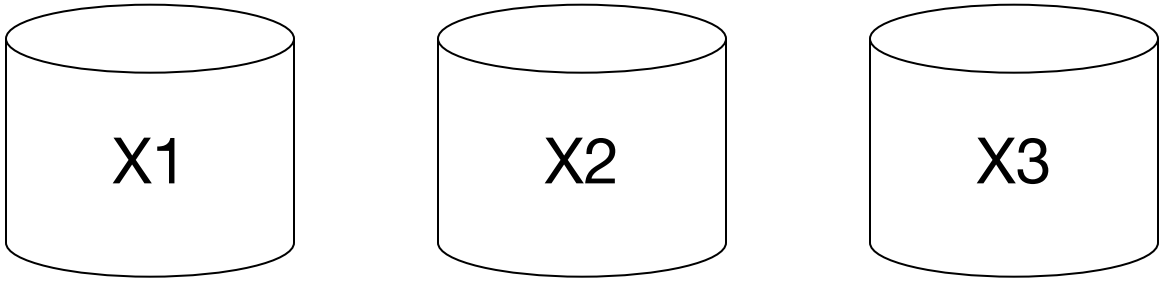
Solution: Make copies of data!

Naïve Way: Redundant Storage

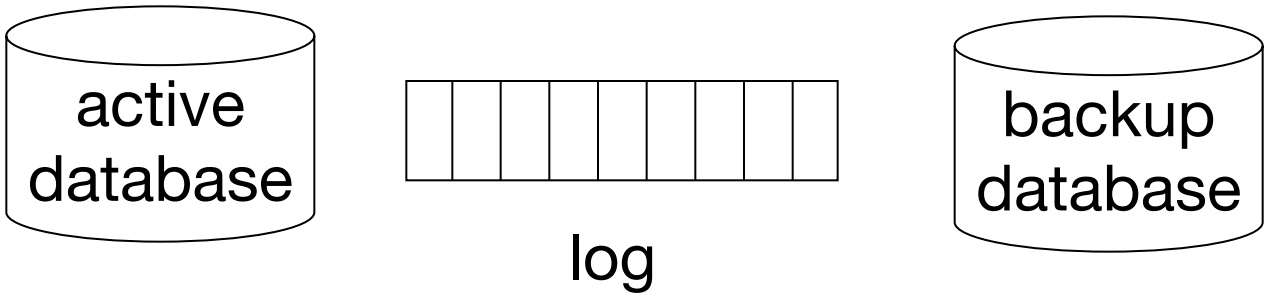
Example: keep 3 copies on separate disks

Output(X) → three outputs

Input(X) → three inputs + vote



Better Way: Log-Based Backup



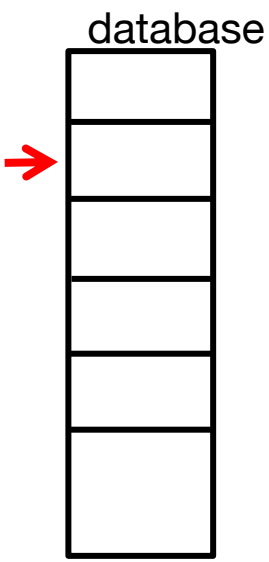
If active database is lost,

- restore active database from backup
- bring up-to-date using redo entries in log

Backup Database

Just like a checkpoint, except that we write the full database

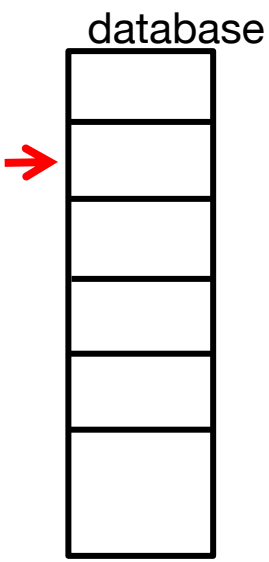
```
create backup database:

for i := 1 to DB_Size do
   [read DB block i; write to backup]

[transactions run concurrently]
```



Backup Database

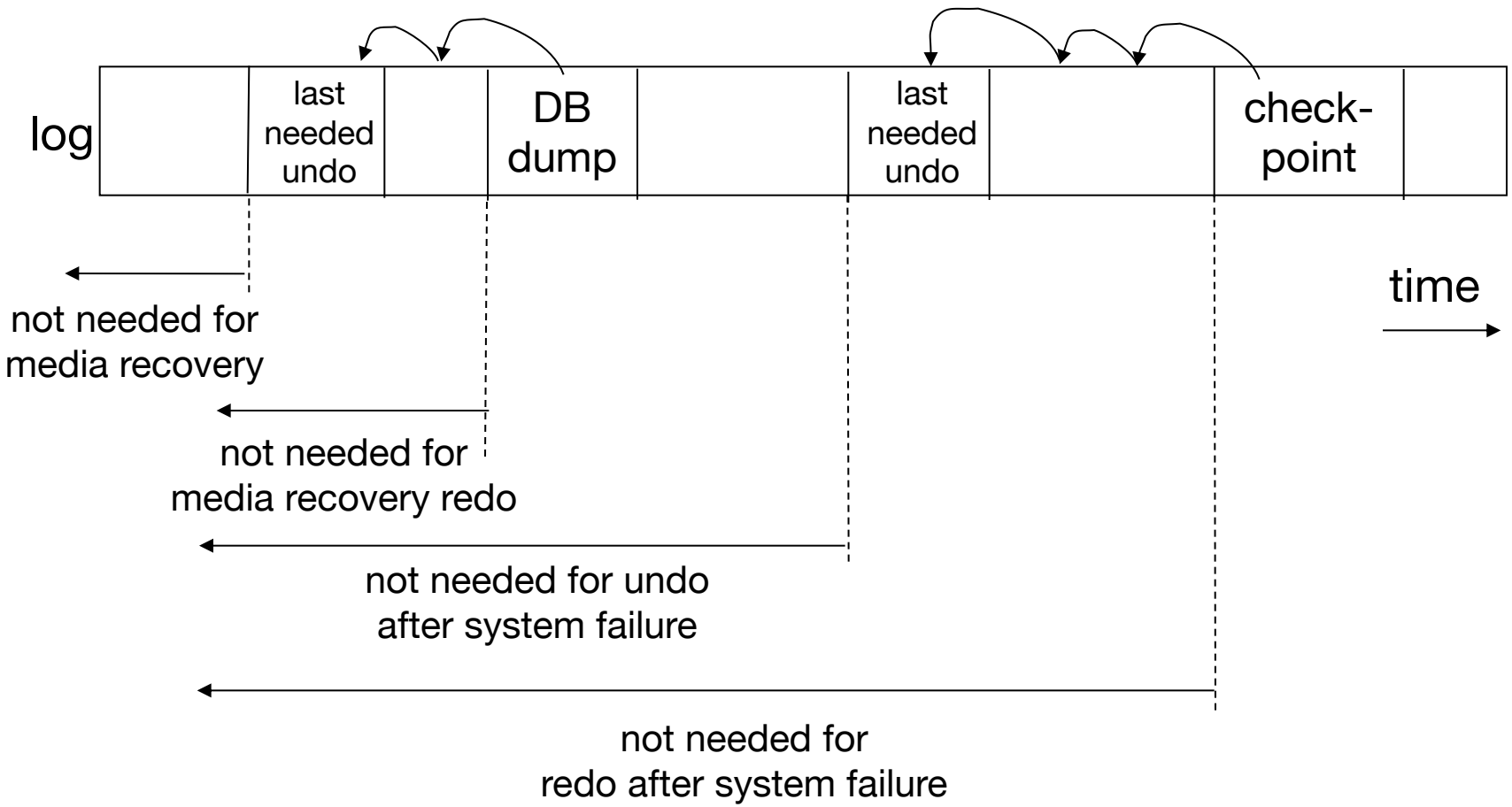
Just like a checkpoint, except that we write the full database

```
create backup database:

for i := 1 to DB_Size do
   [read DB block i; write to backup]

[transactions run concurrently]
```



Restore from backup DB and log:
Similar to recovery from checkpoint and log

When Can Logs Be Discarded?



Summary

Consistency of data: maintain constraints

One source of problems: failures

» Logging
» Redundancy

Another source of problems: data sharing

» We'll cover this next!