故障恢复

徐辰
华东师范大学
数据科学与工程学院
cxu@dase.ecnu.edu.cn

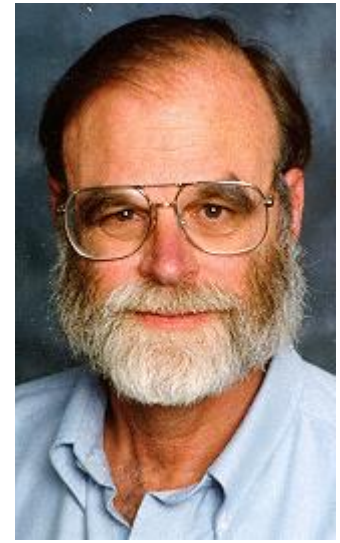
关系数据库三大成就

- 关系模型（数据模型）
- 查询优化（查询处理）
- 事务处理（事务管理）

事务管理面对的问题：数据正确性

- 硬件失效
  - 宕机/停电
  - 硬件损坏
  - 灾难
- 软件错误
  - Bug
  - 恶意攻击
- 并发问题
  - 多个用户同时更新数据出现异常

事务的概念



- ACID
  - 原子性（Atomicity）
    - 一个事务（transaction）要么没有开始，要么全部完成，不存在中间状态。
  - 一致性（Consistency）
    - 事务的执行不会破坏数据的正确性，即符合约束。
  - 隔离性（Isolation）
    - 多个事务不会相互破坏。
  - 持久性（Durability）
    - 事务一旦提交成功，对数据的修改不会丢失。

故障恢复

例子：一个 简单的事务

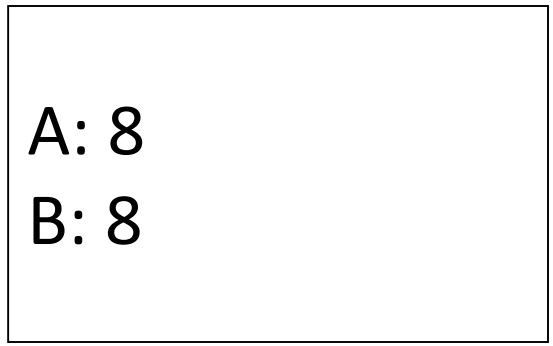
Constraint: A=B

T_1: A $\leftarrow$ A $\times$ 2

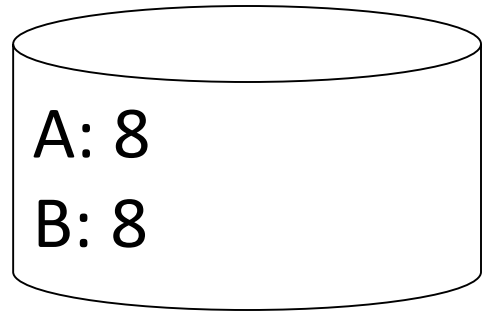
B $\leftarrow$ B $\times$ 2

- Input (x)：将x从磁盘读入内存。
- Output (x)：将x从内存写到磁盘。
- Read (x,t)：事务在内存中读取x。
- Write (x,t)：事务在内存中给x赋值。

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);

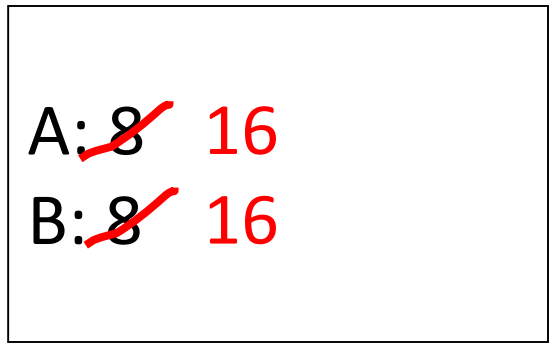


memory

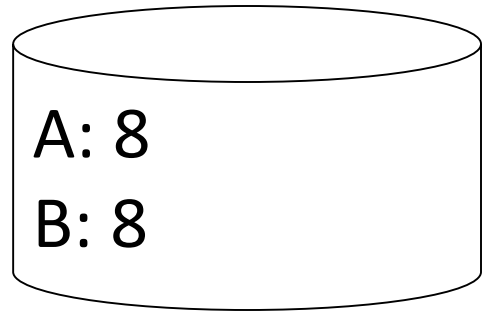


disk

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);



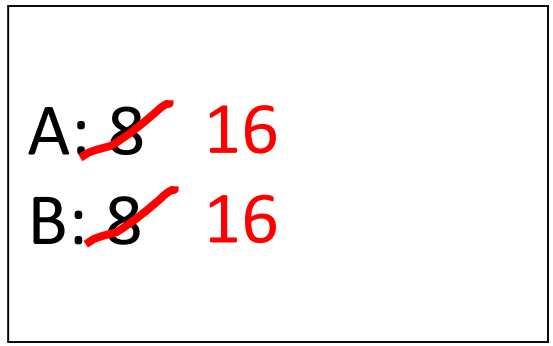
memory



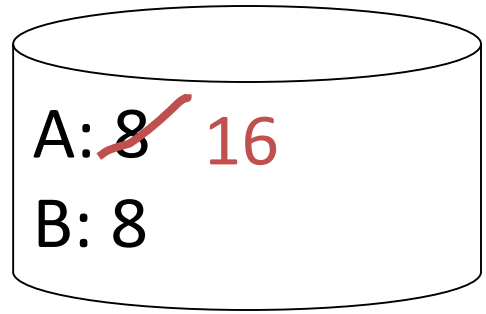
disk

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);

failure!



memory

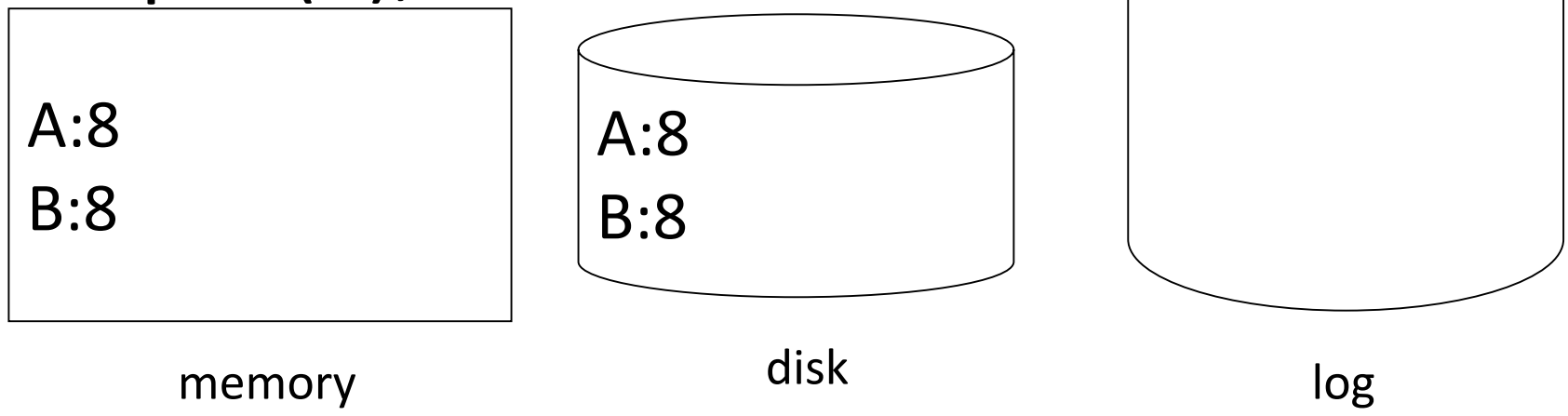


disk

Undo日志

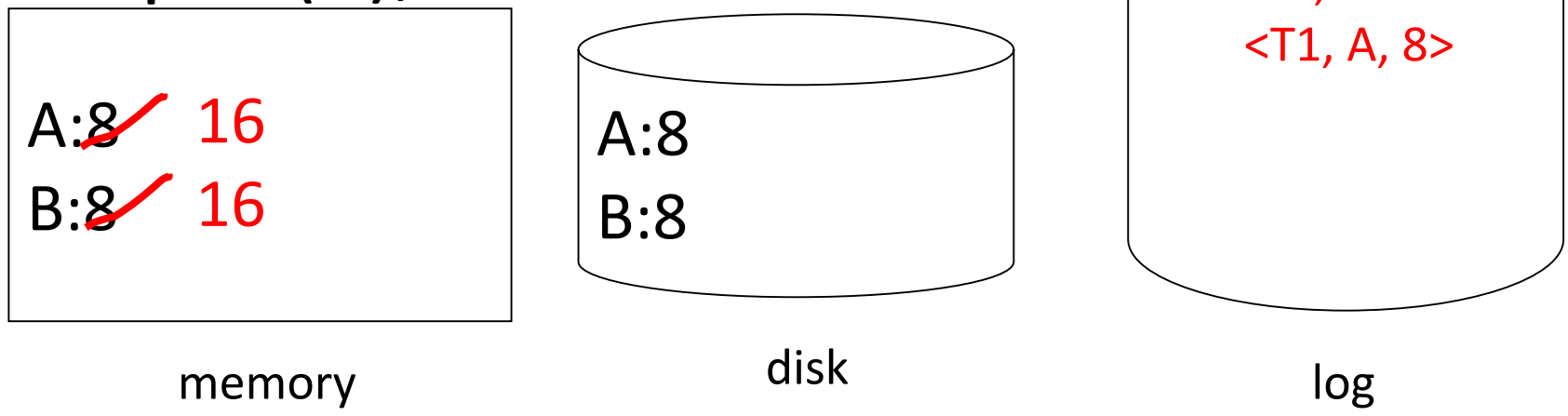
Undo logging (Immediate modification)

T_1: 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);



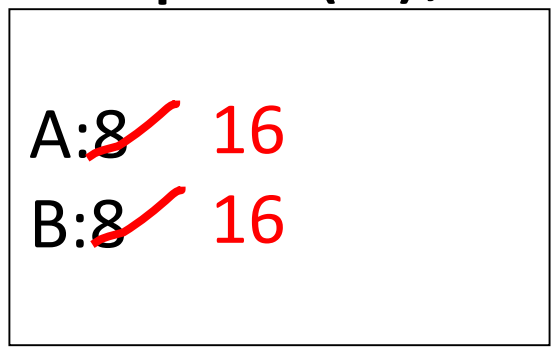
Undo logging (Immediate modification)

T_1: Read (A,t); t ← t×2 A=B
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);

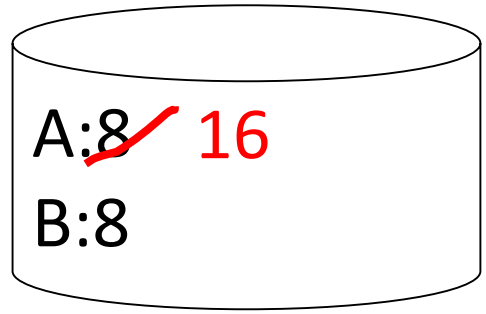


Undo logging (Immediate modification)

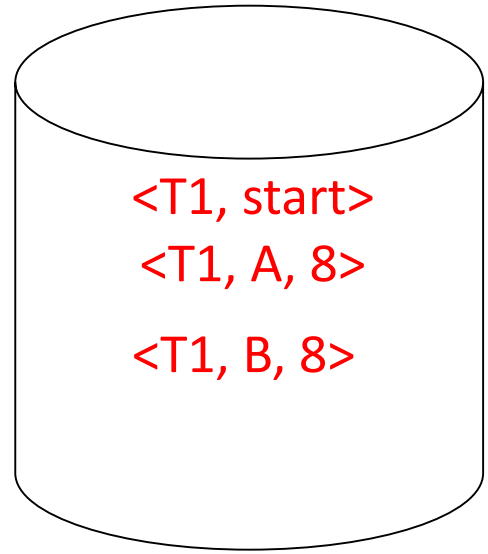
T_1: Read (A,t); t ← t×2 A=B
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);



memory



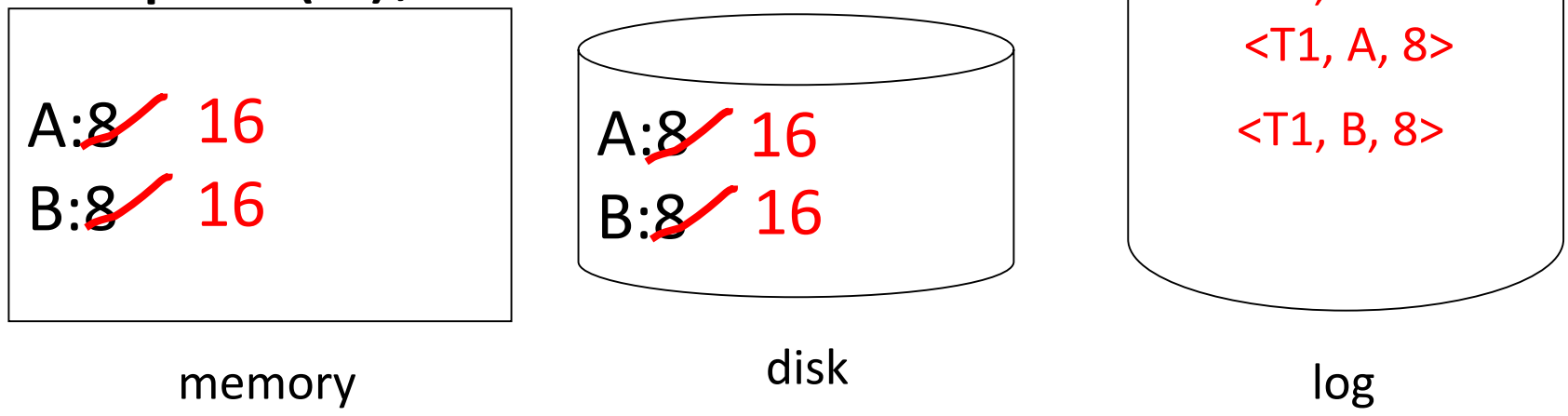
disk



log

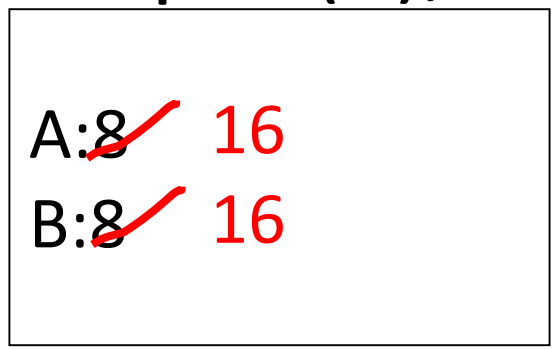
Undo logging (Immediate modification)

T_1: Read (A,t); t ← t×2 A=B
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);

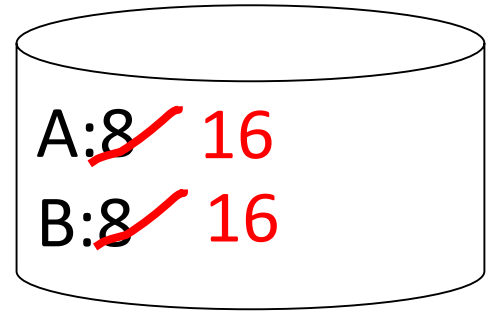


Undo logging (Immediate modification)

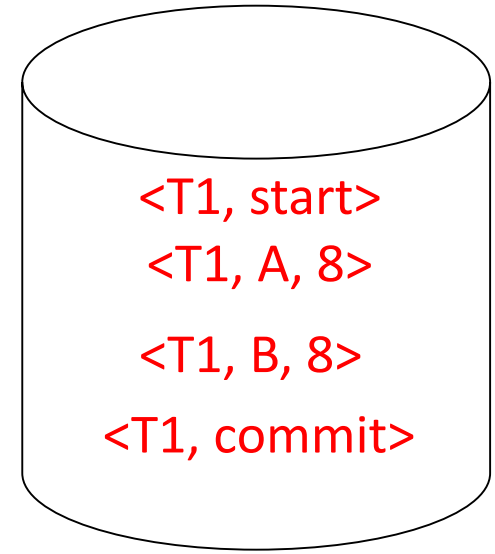
T_1: Read (A,t); t ← t×2 A=B
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);



memory



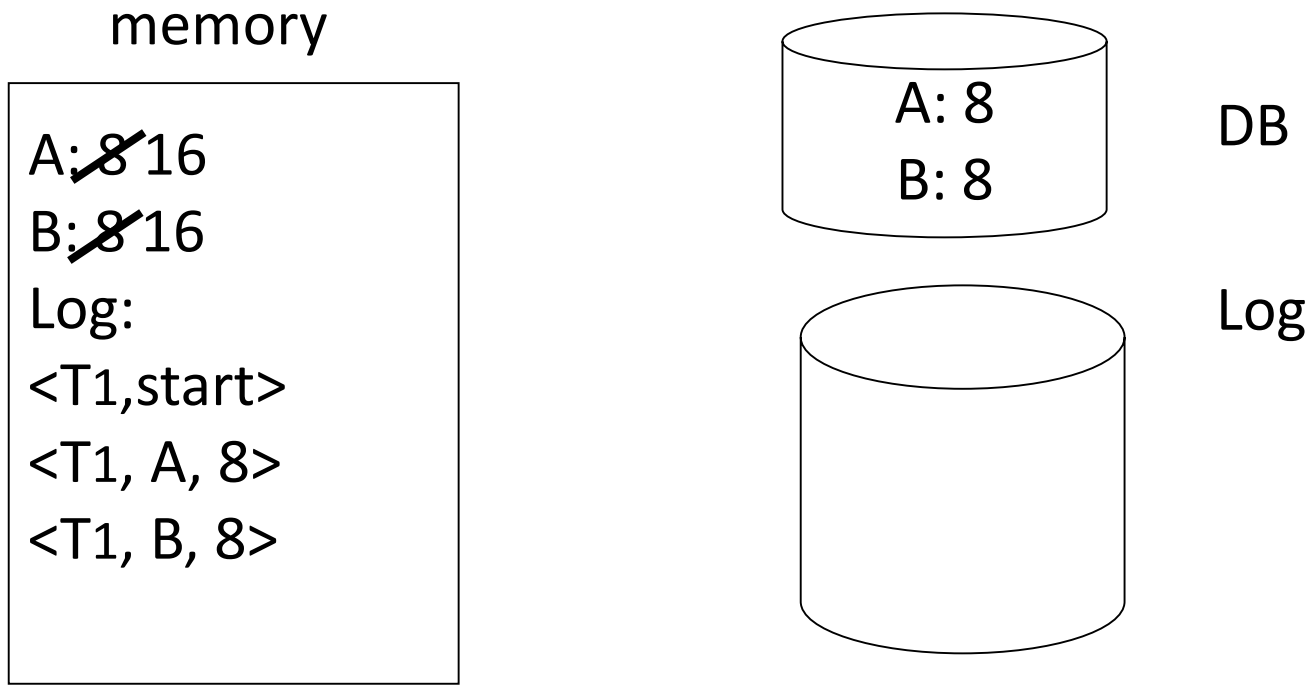
disk



log

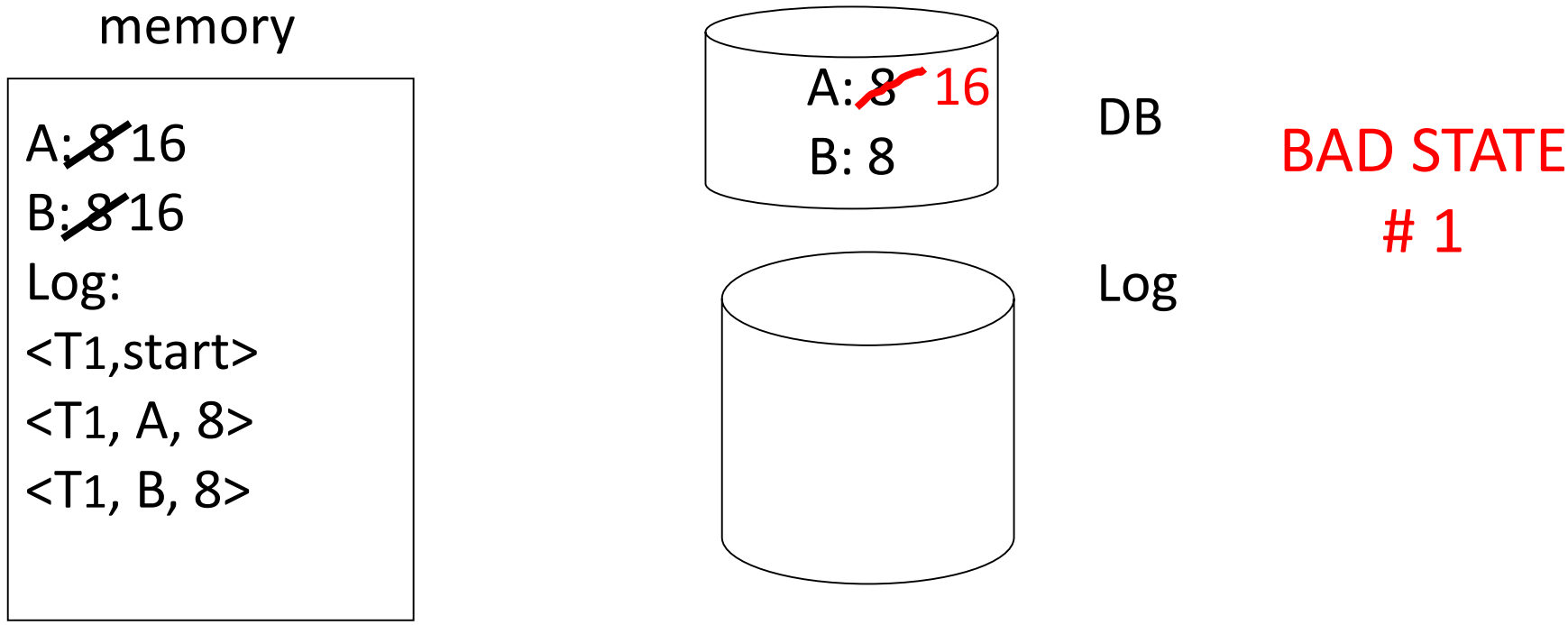
Undo日志需要在数据之前到达磁盘

- Log is first written in memory
- Not written to disk on every action



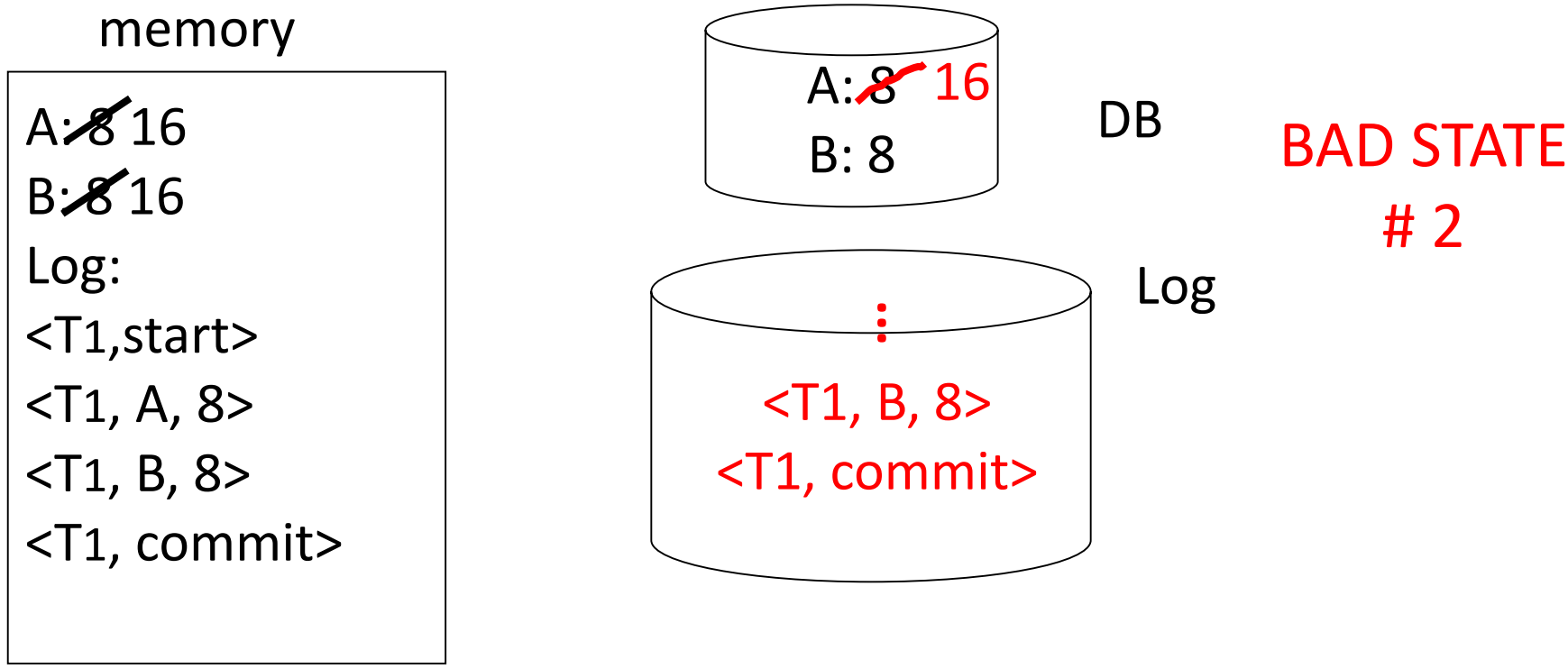
Undo日志需要在数据之前到达磁盘

- Log is first written in memory
- Not written to disk on every action



数据需要在事务提交之前到达磁盘

- Log is first written in memory
- Not written to disk on every action



Undo日志的规则

(1) 每一次对数据的改动都需要记录日志。

(2) 日志记录必须在数据之前到达磁盘。(write ahead logging: WAL)

(3) 事务结束之前，所有的数据和日志必须到达磁盘。（保证持久性）

基于Undo日志的恢复过程

(1) Let S = set of transactions with
<Ti, start> in log, but no <Ti, commit> (or <Ti, abort>)
record in log

(2) For each <Ti, X, v> in log,

in reverse order (latest $\rightarrow$ earliest) do:

if Ti $\in$ S then
- write (X, v)
- output (X)

(3) For each Ti $\in$ S do

write <Ti, abort> to log

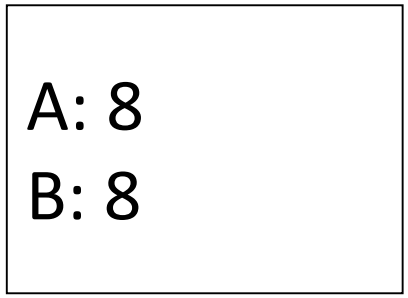
Redo日志

Redo logging (deferred modification)

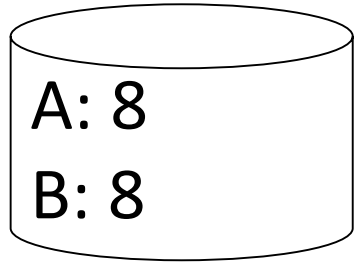
T_1: Read(A,t); t←t×2; write (A,t);

Read(B,t); t←t×2; write (B,t);

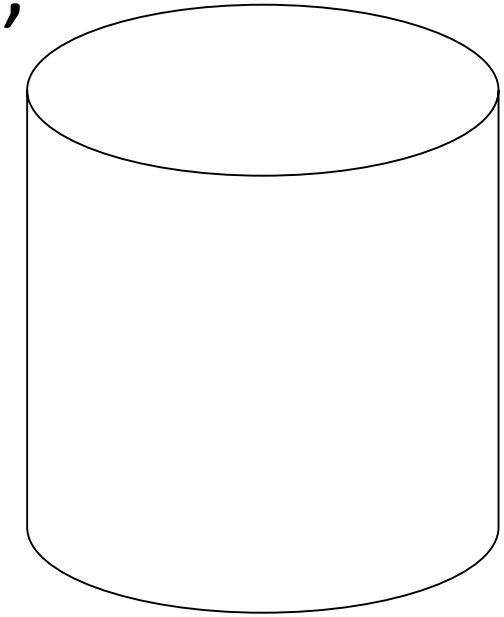
Output(A); Output(B)



memory



DB



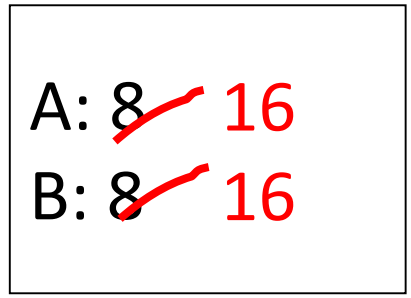
LOG

Redo logging (deferred modification)

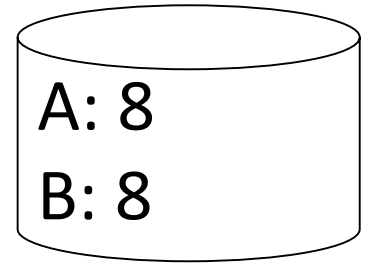
T_1: Read(A,t); t←t×2; write (A,t);

Read(B,t); t←t×2; write (B,t);

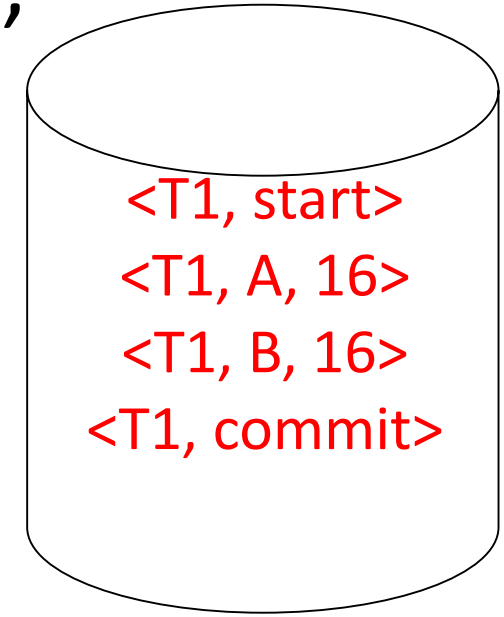
Output(A); Output(B)



memory



DB



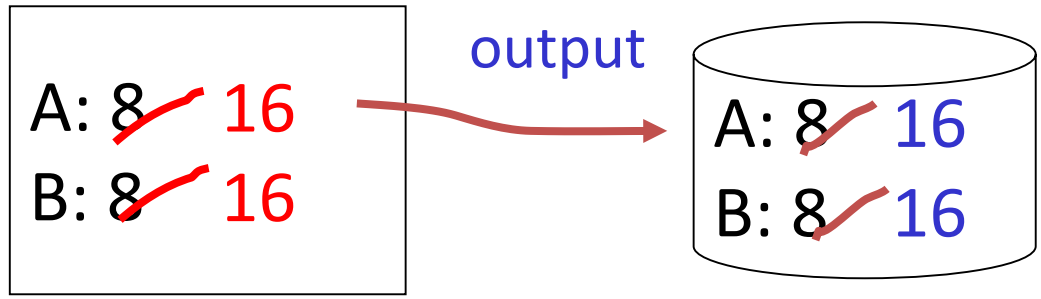
LOG

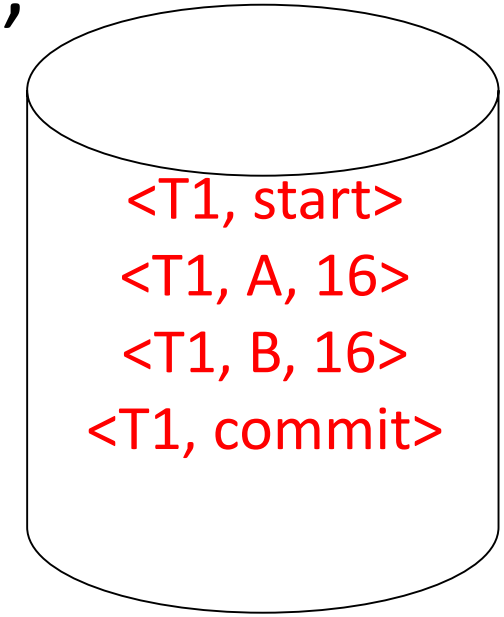
Redo logging (deferred modification)

T_1: Read(A,t); t←t×2; write (A,t);

Read(B,t); t←t×2; write (B,t);

Output(A); Output(B)



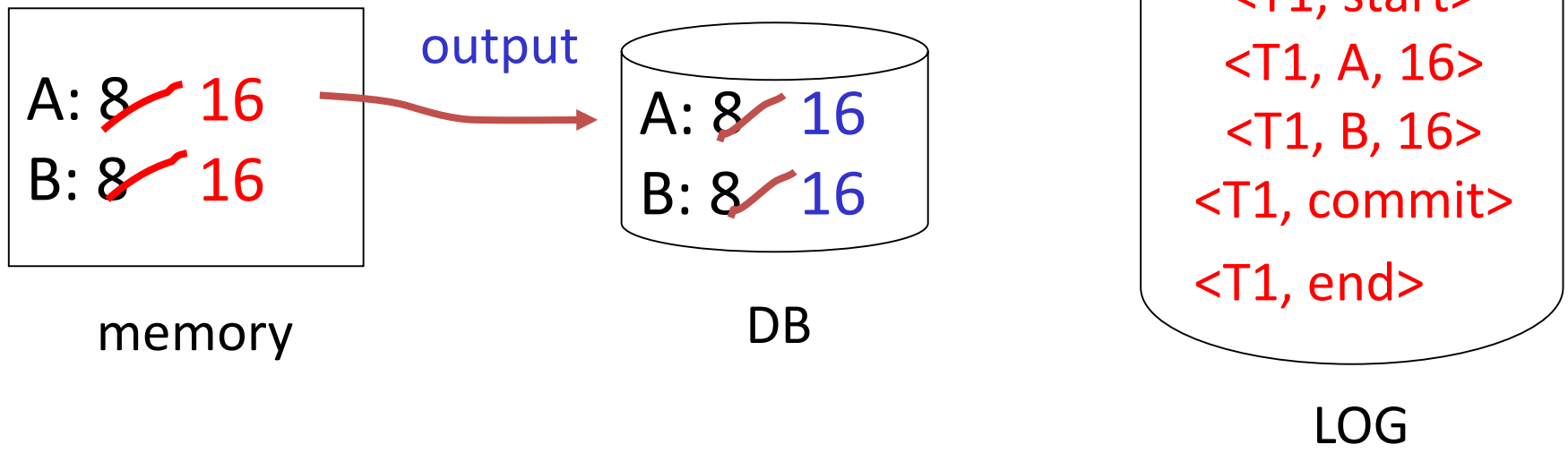


Redo logging (deferred modification)

T_1: Read(A,t); t←t×2; write (A,t);

Read(B,t); t←t×2; write (B,t);

Output(A); Output(B)



Redo日志的规则

(1) 每一次对数据的改动都需要记录日志。

(2) 事务提交之前所有的日志必须到达磁盘。

(3) 事务提交之后才能将数据写到磁盘。

(4) 数据到达磁盘后，需在日志中记录END。

基于Redo日志的恢复过程

(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 { Write(X, v)
Output(X)

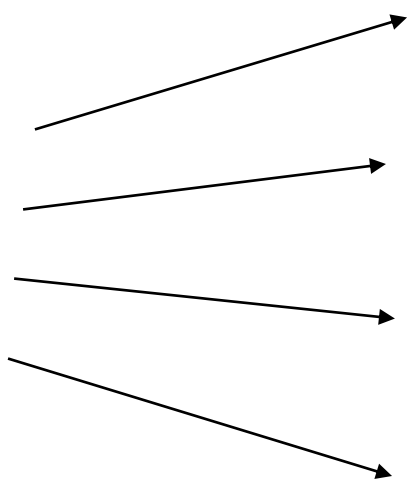
(3) For each Ti $\in$ S

write <Ti, end>

合并多个事务的END

- Want to delay DB flushes for hot objects

Say X :
T1: … update X…
T2: … update X…
T3: … update X…
T4: … update X…

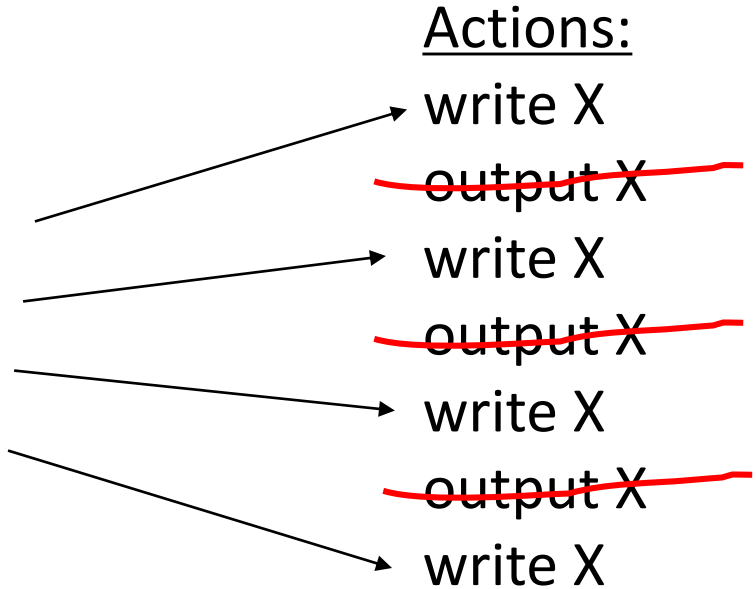


Actions:
write X
output X
write X
output X
write X
output X
write X
output X

合并多个事务的END

- Want to delay DB flushes for hot objects

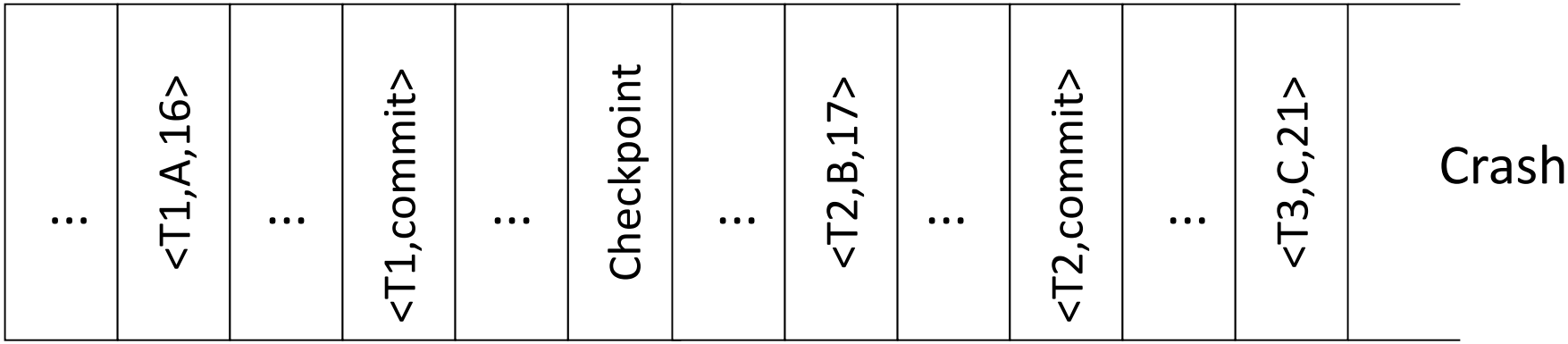
Say X :
T1: … update X…
T2: … update X…
T3: … update X…
T4: … update X…



combined <end> (checkpoint)

检查点（Checkpoint）

Redo log (disk):



Undo和Redo日志的缺点

- Undo日志
  - 事务提交之前必须将数据刷盘。
- Redo日志
  - 事务提交之前不能将数据刷盘。

更好的方式：Undo/Redo日志

Update $\Rightarrow$ <Ti, Xid, New X val, Old X val>

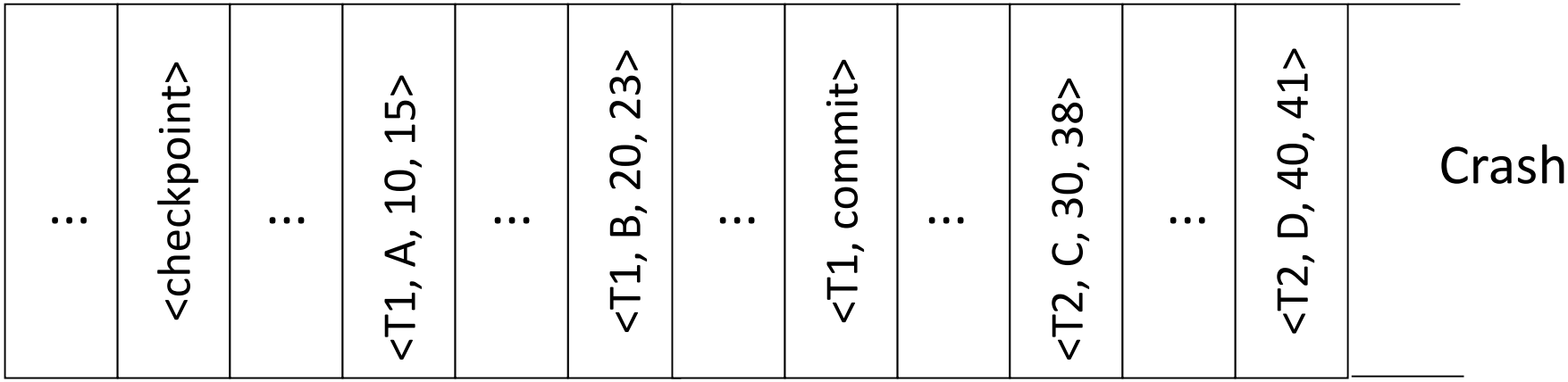
page X

Undo/Redo日志的规则

- 数据可以在任何时间到达磁盘。（事务提交前或事务提交后）
- 日志记录必须在数据之前到达磁盘。(write ahead logging: WAL)
- 事务提交之前所有的日志必须到达磁盘。

例子：Undo/Redo日志

log (disk):



现代数据库的日志模式：ARIES

- 一种Undo/Redo日志
  - ARIES algorithms, developed by C.Mohan at IBM Almaden in the early 90’s
  - http://www.almaden.ibm.com/u/mohan/ARIES_Impact.htm