

CSE 562 Database Systems

Failure Recovery

Some slides are based or modified from originals by
Database Systems: The Complete Book,
Pearson, Prentice Hall 2nd Edition,
©2006 Garcia-Molina, Ullman, and Widom

cse@buffalo

Integrity or Correctness of Data

- Would like data to be “accurate” or “correct” at all times

EMP	Name	Age
	White	52
	Green	3421
	Gray	1

Integrity or Consistency Constraints

- Predicates data must satisfy
- Examples:
 - x is key of relation R
 - $x \rightarrow y$ holds in R
 - $\text{Domain}(x) = \{\text{Red}, \text{Blue}, \text{Green}\}$
 - α is valid index for attribute x of R
 - no employee should make more than twice the average salary

Definition

- Consistent state: satisfies all constraints
- Consistent DB: DB in consistent state

Constraints (as we use here) may not capture “full correctness”

Example 1: Transaction constraints

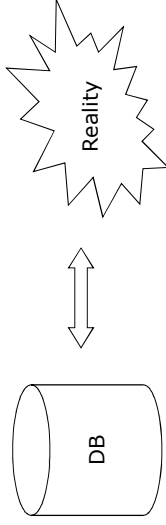
- When salary is updated,
new salary > old salary
- When account record is deleted,
balance = 0

Note: could be “emulated” by simple constraints, e.g.,

account	Acct #	...	balance	deleted?
---------	--------	-----	---------	----------

Constraints (as we use here) may not capture “full correctness”

Example 2: Database should reflect real world



In any case, continue with constraints...

Observation: DB cannot be consistent always!

Example: $a_1 + a_2 + \dots + a_n = \text{TOT}$ (constraint)

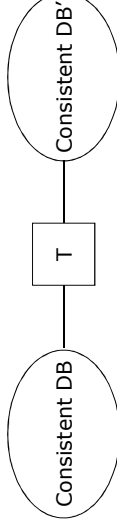
Deposit \$100 in a_2 : $\left\{ \begin{array}{l} a_2 \leftarrow a_2 + 100 \\ \text{TOT} \leftarrow \text{TOT} + 100 \end{array} \right.$

Example: $a_1 + a_2 + \dots + a_n = \text{TOT}$ (constraint)

Deposit \$100 in a_2 : $\begin{cases} a_2 \leftarrow a_2 + 100 \\ \text{TOT} \leftarrow \text{TOT} + 100 \end{cases}$

a_2	:	50	→	:	150	→	:	150
	:	:		:	:		:	:
	TOT	1000		TOT	1000		TOT	1100

Transaction: collection of actions that preserve consistency



Big Assumption

If T starts with consistent state +
 T executes in isolation
 \Rightarrow T leaves consistent state

Correctness (informally)

- If we stop running transactions, DB left consistent
- Each transaction sees a consistent DB

How Can Constraints Be Violated?

- Transaction bug
- DBMS bug
- Hardware failure
 - e.g., disk crash alters balance of account
- Data sharing, e.g.:
 - T1: give 10% raise to programmers
 - T2: change programmers ⇒ systems analysts

UB CSE 562 Spring 2009

13

How Can We Prevent/Fix Violations?

- Chapter 17: Due to failures only
- Chapter 18: Due to data sharing only
- Chapter 19: Due to failures and sharing

UB CSE 562 Spring 2009

14

Will Not Consider

- How to write correct transactions
- How to write correct DBMS
- Constraint checking & repair
 - That is, solutions studied here do not need to know constraints

UB CSE 562 Spring 2009

15

Chapter 17: Recovery

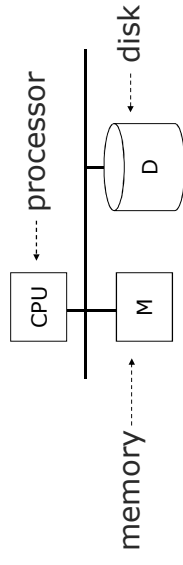
- First order of business:
Failure Model

UB CSE 562 Spring 2009

16

Events — Desired
 — Undesired — Expected
 — Unexpected

Our Failure Model



Desired events: see product manuals....

Undesired expected events:

System crash

- memory lost
- cpu halts, resets

————— that's it!!

Undesired Unexpected: Everything else!

Undesired Unexpected: Everything else!

Examples:

- Disk data is lost
- Memory lost without CPU halt
- CPU implodes wiping out universe...

Is This Model Reasonable?

Approach: Add low level checks + redundancy to increase probability model holds

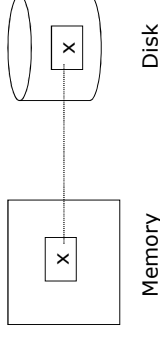
E.g., { Replicate disk storage (stable store)
Memory parity
CPU checks

UB CSE 562 Spring 2009

21

Second Order of Business:

Storage hierarchy



UB CSE 562 Spring 2009

22

Operations

- Input (x): block containing x → memory
- Output (x): block containing x → disk
- Read (x,t): do input(x) if necessary
t ← value of x in block
- Write (x,t): do input(x) if necessary
value of x in block ← t

UB CSE 562 Spring 2009

23

Key Problem: Unfinished Transaction

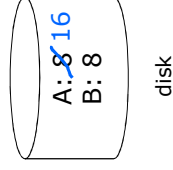
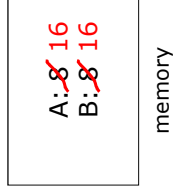
Example

Constraint: $A=B$
T1: $A \leftarrow A \times 2$
 $B \leftarrow B \times 2$

UB CSE 562 Spring 2009

24

T1: Read (A,t); t ← t×2
 Write (A,t);
 Read (B,t); t ← t×2
 Write (B,t);
 Output (A);
 Output (B);



failure!

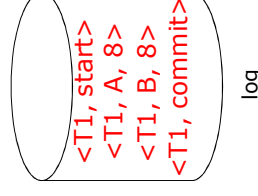
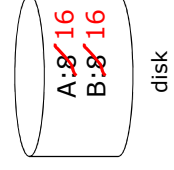
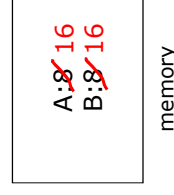
- Need atomicity: execute all actions of a transaction or none at all

One solution: undo logging
 (immediate modification)

due to: Hansel and Gretel, 782 AD

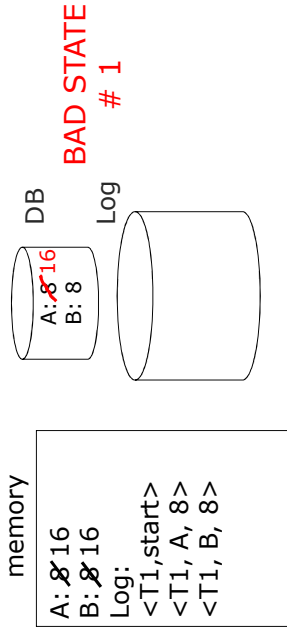
Undo Logging (Immediate modification)

T1: Read (A,t); t ← t×2 A=B
 Write (A,t);
 Read (B,t); t ← t×2
 Write (B,t);
 Output (A);
 Output (B);



One “Complication”

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

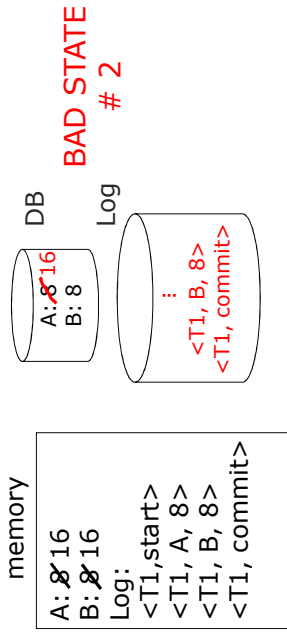


UB CSE 562 Spring 2009

29

One “Complication”

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



UB CSE 562 Spring 2009

30

Undo Logging Rules

- (1) For every action generate undo log record (containing old value)
- (2) Before x is modified on disk, log records pertaining to x must be on disk (write ahead logging: WAL)
- (3) Before commit is flushed to log, all writes of transaction must be reflected on disk

UB CSE 562 Spring 2009

31

Recovery Rules: Undo Logging

- For every T_i with $\langle T_i, \text{start} \rangle$ in log:
 - If $\langle T_i, \text{commit} \rangle$ or $\langle T_i, \text{abort} \rangle$ in log:
 - Do nothing
 - Else
 - For all $\langle T_i, X, v \rangle$ in log:
 - write (X, v)
 - output (X)
 - Write $\langle T_i, \text{abort} \rangle$ to log

IS THIS CORRECT??

UB CSE 562 Spring 2009

32

Recovery Rules: Undo Logging

- (1) Let S = set of transactions with $\langle T_i, \text{start} \rangle$ in log, but no $\langle T_i, \text{commit} \rangle$ (or $\langle T_i, \text{abort} \rangle$) record in log
- (2) For each $\langle T_i, X, v \rangle$ in log, in reverse order (latest \rightarrow earliest) do:
 - if $T_i \in S$ then $\left\{ \begin{array}{l} \text{- write } (X, v) \\ \text{- output } (X) \end{array} \right.$
- (3) For each $T_i \in S$ do
 - write $\langle T_i, \text{abort} \rangle$ to log

UB CSE 562 Spring 2009

33

Question

- Can writes of $\langle T_i, \text{abort} \rangle$ records be done in any order (in Step 3)?
 - **Example:** T_1 and T_2 both write A
 - T_1 executed before T_2
 - T_1 and T_2 both rolled-back
 - $\langle T_1, \text{abort} \rangle$ written but NOT $\langle T_2, \text{abort} \rangle$



UB CSE 562 Spring 2009

34

What if failure during recovery?

No problem! Undo **idempotent!**

UB CSE 562 Spring 2009

35

To Discuss:

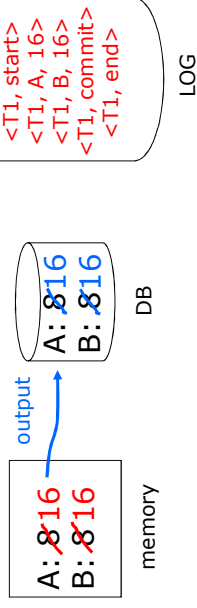
- Redo logging
- Undo/redo logging, why both?
- Real world actions
- Checkpoints
- Media failures

UB CSE 562 Spring 2009

36

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)



UB CSE 562 Spring 2009

37

Redo Logging Rules

- (1) For every action, generate redo log record (containing new value)
- (2) Before X is modified on disk (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 flushed to disk

UB CSE 562 Spring 2009

38

Recovery Rules: Redo Logging

- For every T_i with $\langle T_i, \text{commit} \rangle$ in log:
 - For all $\langle T_i, X, v \rangle$ in log:
 - { Write(X, v)
 - Output(X)

IS THIS CORRECT??

UB CSE 562 Spring 2009

39

Recovery Rules: Redo Logging

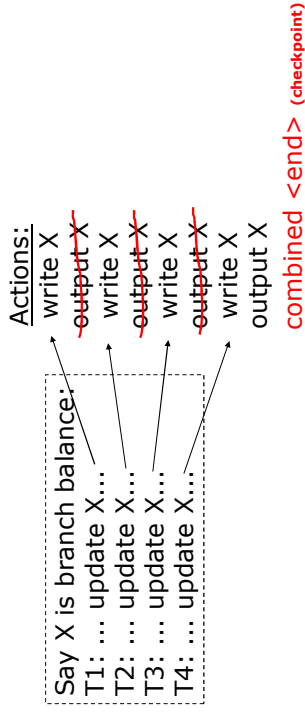
- (1) Let S = set of transactions with $\langle T_i, \text{commit} \rangle$ (and no $\langle T_i, \text{end} \rangle$) in log
- (2) For each $\langle T_i, X, v \rangle$ in log, in forward order (earliest \rightarrow latest) do:
 - If $T_i \in S$ then { Write(X, v)
 - Output(X)
- (3) For each $T_i \in S$, write $\langle T_i, \text{end} \rangle$

UB CSE 562 Spring 2009

40

Combining <Ti, end> Records

- Want to delay DB flushes for hot objects



Solution: Checkpoint

Periodically:

- no <ti, end> actions
- simple checkpoint

- Do not accept new transactions
- Wait until all transactions finish
- Flush all log records to disk (log)
- Flush all buffers to disk (DB) (do not discard buffers)
- Write "checkpoint" record on disk (log)
- Resume transaction processing

Example: What To Do At Recovery?

Redo log (disk):

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

Key Drawbacks

- Undo logging: cannot bring backup DB copies up to date
- Redo logging: need to keep all modified blocks in memory until commit

Solution: Undo/Redo Logging!

Update \Rightarrow $\langle T_i, Xid, \text{New } X \text{ val}, \text{Old } X \text{ val} \rangle$
page X

UB CSE 562 Spring 2009

45

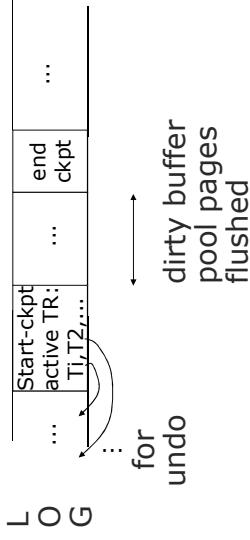
Rules

- Page X can be flushed before or after T_i commit
- Log record flushed before corresponding updated page (WAL)
- Flush at commit (log only)

UB CSE 562 Spring 2009

46

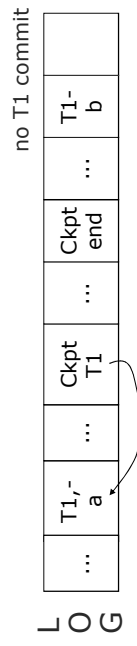
Non-Quiesce Checkpoint



UB CSE 562 Spring 2009

47

Example: What To Do At Recovery Time?

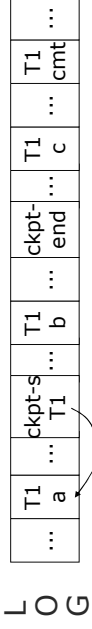


Undo T_1 (undo a,b)

UB CSE 562 Spring 2009

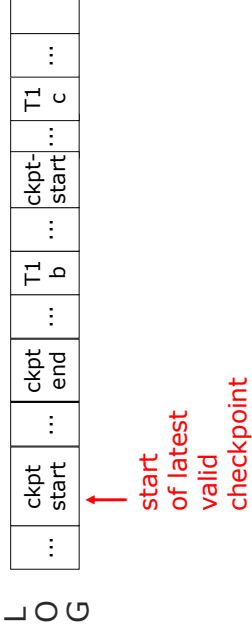
48

Example



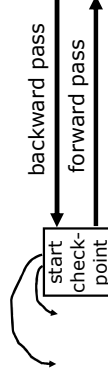
Redo T1 (redo b,c)

Recover From Valid Checkpoint



Recovery Process

- **Backwards pass**
(end of log \Rightarrow 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 active list) - S
- **Forward pass** (latest checkpoint start \Rightarrow end of log)
 - redo actions of S transactions



Real World Actions

E.g., dispense cash at ATM
 $T_i = a_1 a_2 \dots a_j \dots a_n$

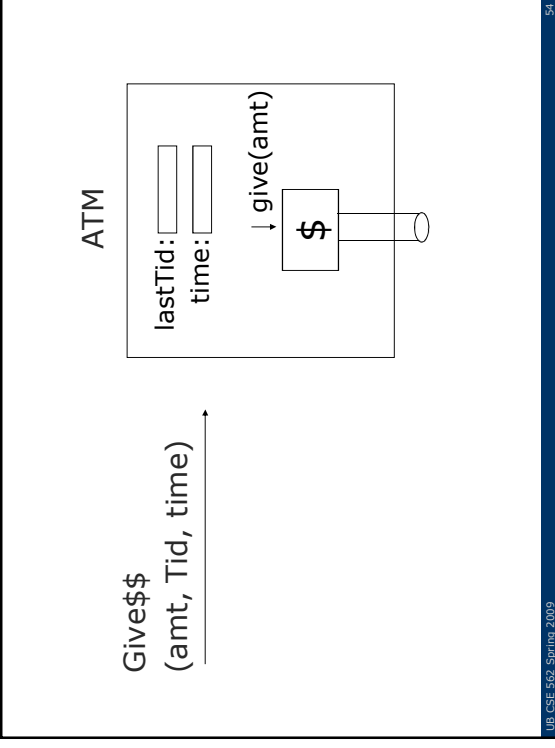
↓
\$

Solution

- (1) execute real-world actions after commit
- (2) try to make idempotent

UB CSE 562 Spring 2009

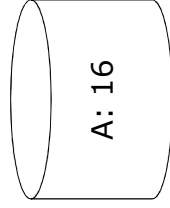
53



UB CSE 562 Spring 2009

54

Media Failure (loss of non-volatile storage)



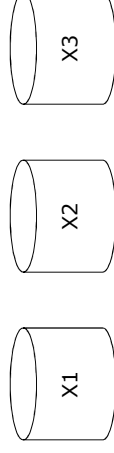
Solution: Make copies of data!

UB CSE 562 Spring 2009

55

Example 1: Triple Modular Redundancy

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



UB CSE 562 Spring 2009

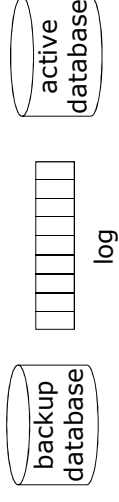
56

Example 2: Redundant Writes, Single Reads

- Keep N copies on separate disks
- Output(X) --> N outputs
- Input(X) --> Input one copy
 - if ok, done
 - else try another one

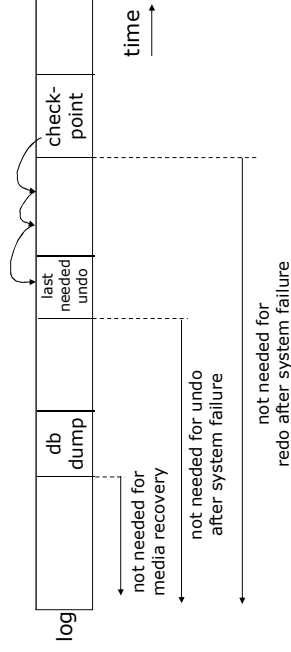
Assumes bad data can be detected

Example 3: DB Dump + Log



- If active database is lost,
 - restore active database from backup
 - bring up-to-date using redo entries in log

When Can Log Be Discarded?



Summary

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