ARIES: A Transaction Recovery Method

Slides by Jessica Wong
(Modified from George Tsiknis' CPSC 304 slides and Ramakrishnan and Gehrke's slides from "Database Management Systems")

Discussion by Sampoorna Biswas

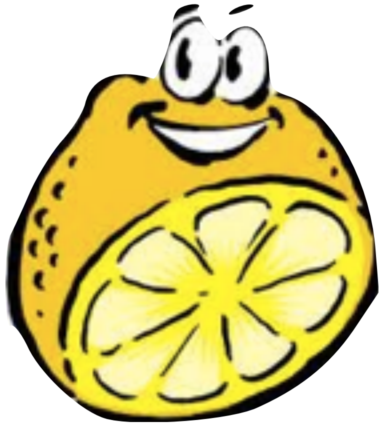
ARIES

What did we read?

- Super long survey paper
- Basically covers all the intricate details of ARIES
- Gives an idea of how complicated it is to implement a system that allows for transaction rollback and recovery

Review: Transactions

- A transaction is a set of read and write operations that are executed as one unit
- One of four states:
  - **Active**: making progress
  - **Failed**: cannot continue due to some type of error
  - **Aborted**: DB had to roll back to a previous save point
  - **Committed**: finished without running into an error



Review: ACID

- **Atomicity**: Either all actions in the transaction occur, or none occur.

- **Consistency**: If each transaction is consistent, and the DB starts in a consistent state, then the DB ends up being consistent.

- **Isolation**: The execution of one transaction is isolated from that of other transactions.

- **Durability**: If a transaction commits, then its effects persist.

Buffer Pool Management

- Data objects are stored on pages in a database (stable storage)
- Whenever an object is modified, the page that it is on needs to be fetched to the memory buffer
  - Page becomes **dirty**
- Save values by writing dirty pages to disk

Buffer Management Policy

- **Steal**: Write pages to disk and take the buffer regardless of transaction state

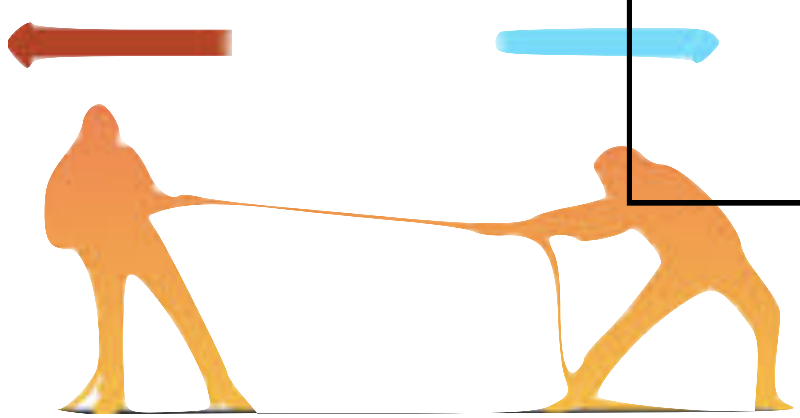
- **No-steal**: Keep a page in memory if it has been updated by an active transaction

- **Force**: Write all the pages modified by a transaction to disk when the transaction commits

- **No-force**: Write pages to disk when buffer space is needed; doesn't care about when a transaction commits

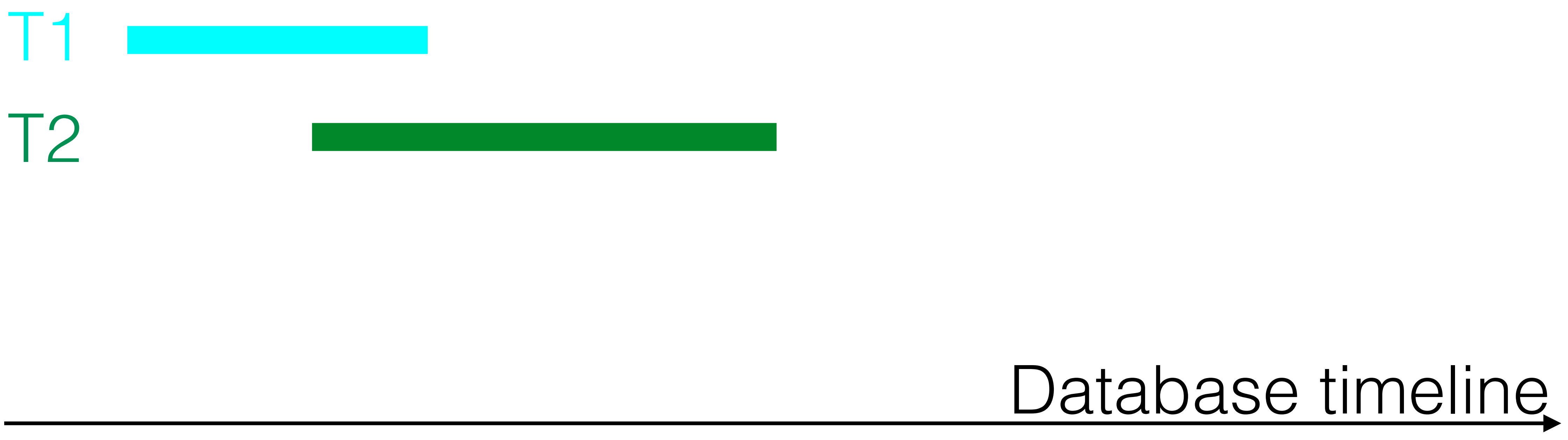
Buffer Management Policy

	No Steal	Steal
Force	Trivial for A & D	
No Force		Difficult for A & D

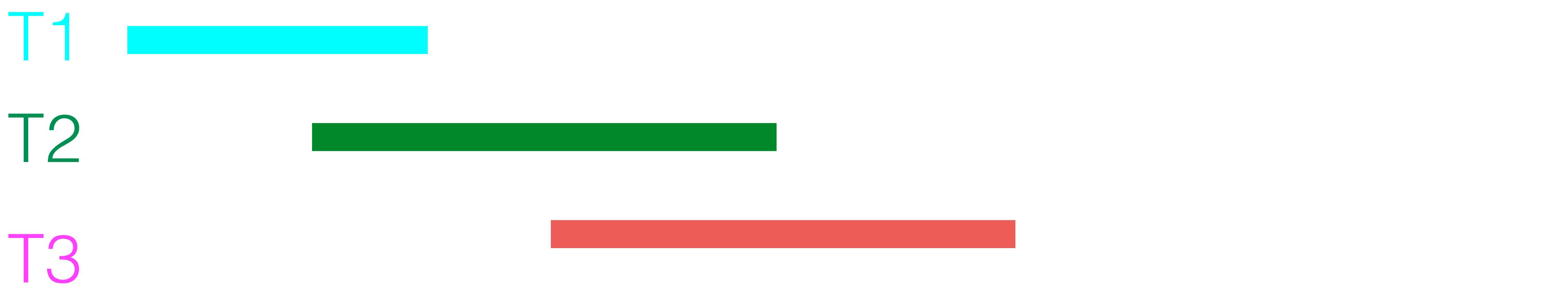


What is it that we need?

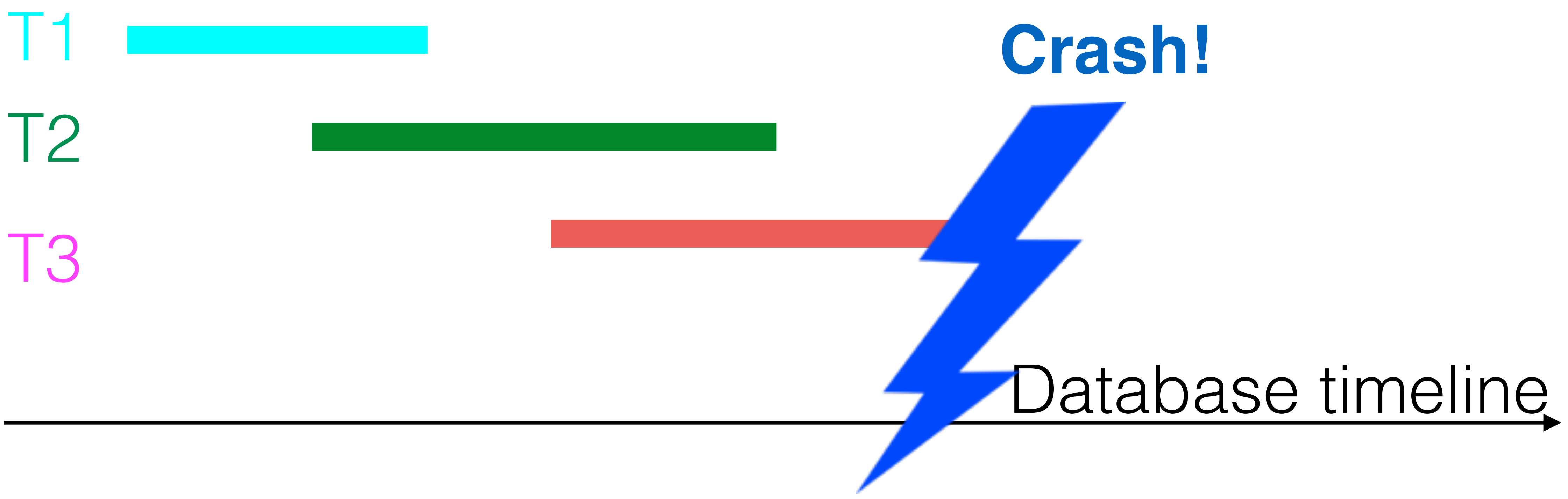
What is it that we need?



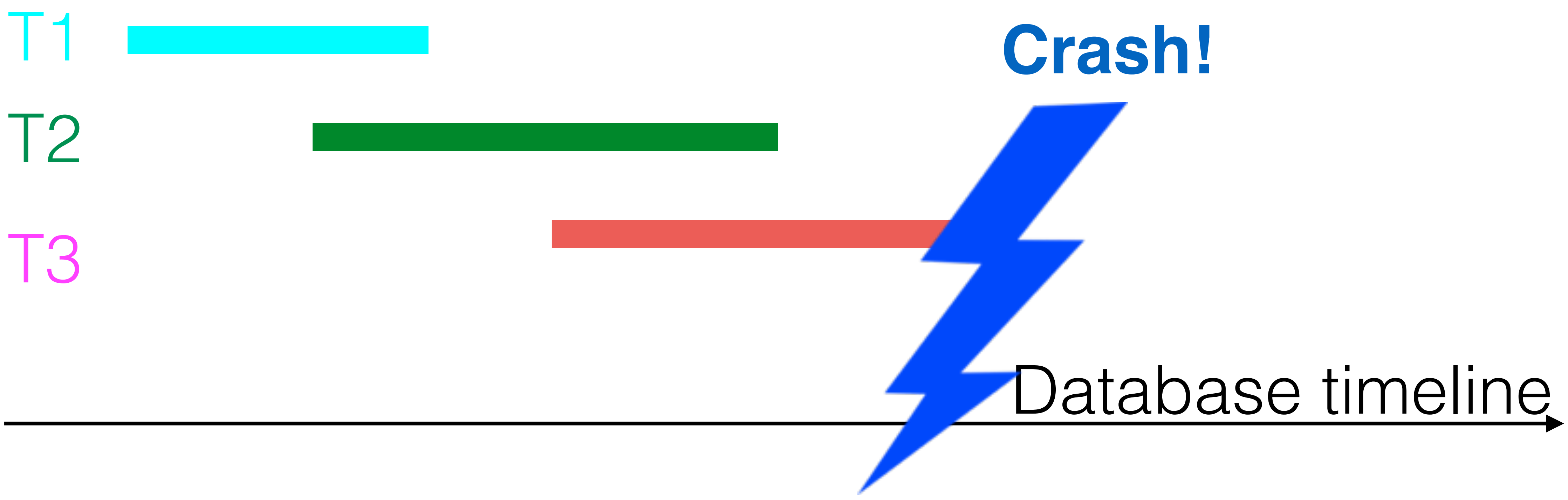
What is it that we need?



What is it that we need?



What is it that we need?



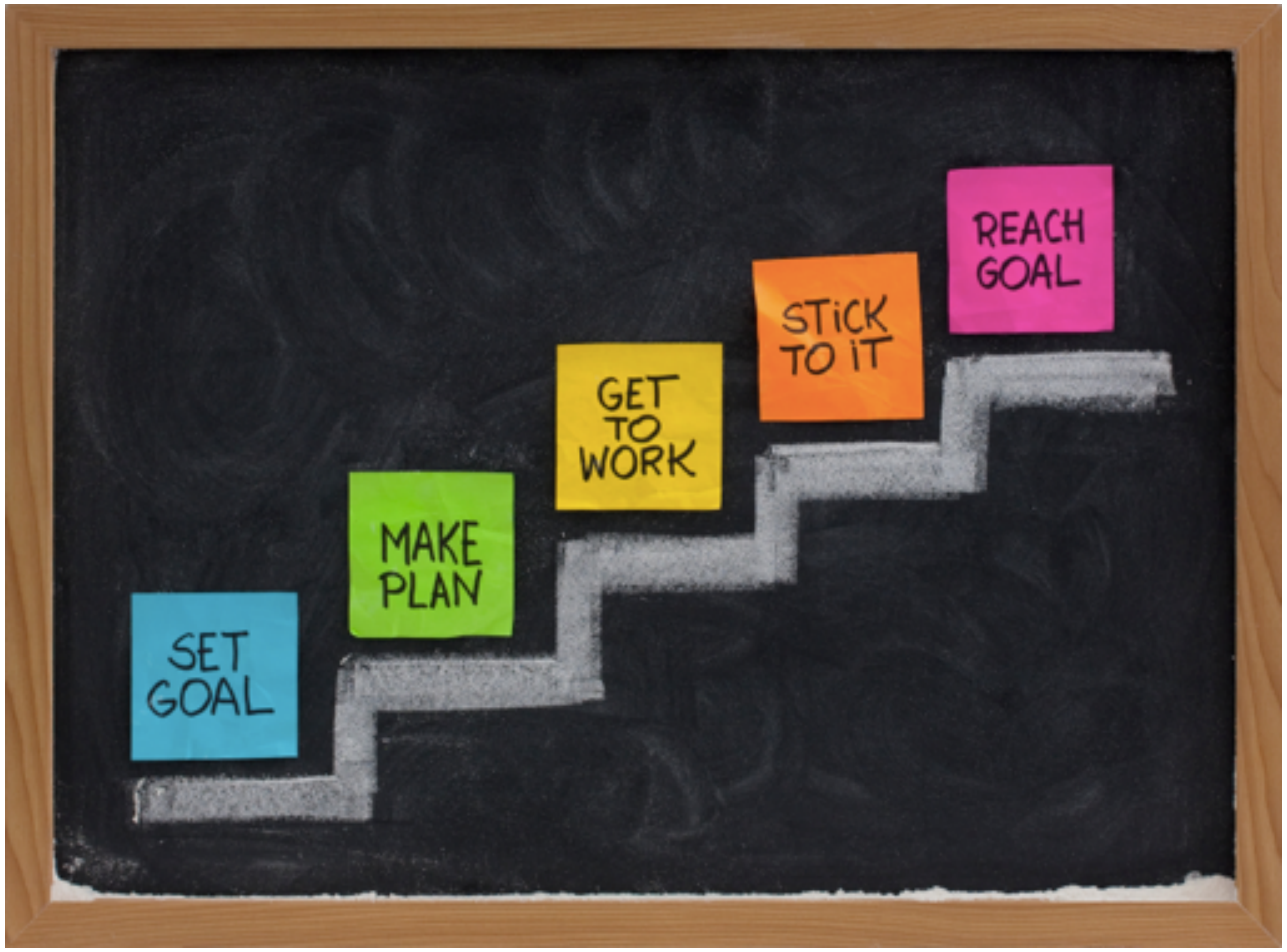
What do we want the state of the server to look like after restart?

ARIES

- Algorithm for Recovery and Isolation Exploiting Semantics
- Algorithm used for database recovery after a database crash
- Created based on nine goals

ARIES Goals

- Simplicity
- Operation Logging
- Flexible storage management
- Partial rollbacks
- Flexible buffer management
- Recovery independence
- Logical undo
- Parallelism and fast recovery
- Minimal overhead



Discussion Question

- **Small groups!**
  - Five minutes group discussion

- Each group randomly picks a piece of paper with ONE Goal on it.
- 1-2 sentences on the ONE Goal your group picks and its challenge
- Vote for the 2 most and 2 least important goals
- Pick one of your votes explain why you voted it as such **OR** add and justify one more goal you think that is important but not listed here.

ARIES: Logging

- Uses a log to keep track of all database changes
  - Record the old and new values of an item for undo/redo purposes
- **WAL** = Write Ahead Logging
  - Forces the writing out of a log record before the corresponding data page gets to disk.
  - Must write all log records for a transaction before commit.

ARIES: Logging

- Uses a log to keep track of all database changes
  - Record the old and new values of an item for undo/redo purposes
- **WAL** = Write Ahead Logging
  - Forces the writing out of a log record before the corresponding data page gets to disk. **=> atomicity**
  - Must write all log records for a transaction before commit.

ARIES: Logging

- Uses a log to keep track of all database changes
  - Record the old and new values of an item for undo/redo purposes
- **WAL** = Write Ahead Logging
  - Forces the writing out of a log record before the corresponding data page gets to disk. **=> atomicity**
  - Must write all log records for a transaction before commit. **=> durability**

ARIES: Logging

- Each log has a unique number, a **log sequence number (LSN)**
  - LSN always increases

- Each data page has a **pageLSN** that records the last LSN number that modified it

ARIES: Logging

- There are different types of logs possible:
  - Update: created during page modification
  - Commit: log forcibly written to stable storage
  - Abort
  - End: created for both aborted and committed transactions
  - Compensation Log Records (CLRs): used during undos

ARIES: Logging

- Different types of log records can hold differing amounts of information:
    - prevLSN
    - transID
    - type
    - pageID
    - length
    - offset
    - before-image
    - after-image
    - undoNextLSN

Update records only

CLR records only

ARIES: Logging

- The **transaction table** stores all active transactions
    - Contains transID, state, LastLSN, UndoNxtLSN
- Transactions removed from table when the transaction has ended

ARIES: Logging

- The **dirty pages table** stores information about dirty buffer pages
  - Each table entry has PageID and RecLSN
    - RecLSN is the LSN of the action that first made the page dirty

ARIES: Checkpoint

- Helps us not have to rollback super far if the system crashes
- Write to the log:
  - begin_checkpoint record
  - end_checkpoint record: stores the current transaction table and dirty page table
- ARIES stores the LSN of the checkpoint record in a safe place **=> fuzzy checkpoint**

ARIES: Crash Recovery

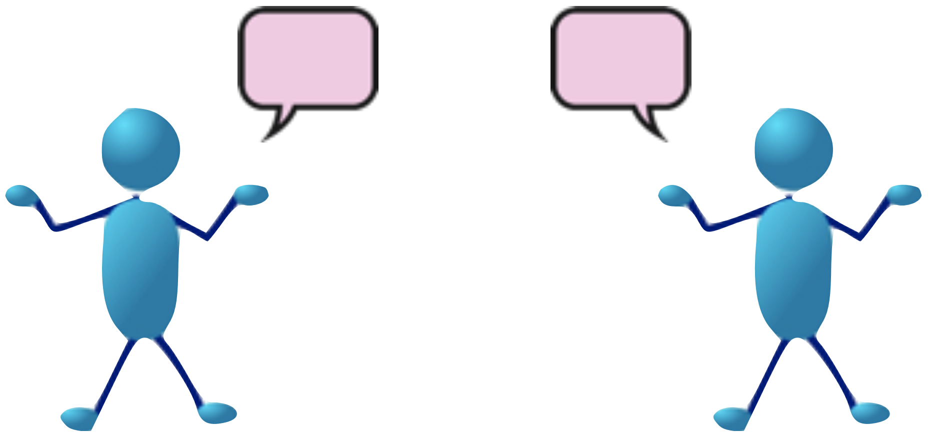
- Three phases:
  - **Analysis**: examine the transactions that have occurred between the crash point and the last check point
  - **Redo**: redo all actions from RecLSN onward
  - **Undo**: undo the effects of the failed transactions from end to first LSN of oldest transaction active at crash time

ARIES: Analysis

- **Goal**: figure out where Redo needs to start at,and what transactions need to be rolled back
- Use the last end_checkpoint record to reconstruct the transaction table and the dirty page table
- Starting from end_checkpoint, update the transaction table and dirty page table according to what the log lists
    - Results in knowing what the smallest LSN in the dirty page table is (i.e., where Redo starts)

Discussion Question

- If you were designing a transaction processing system, would you do a checkpoint after the analysis phase? Why or why not?

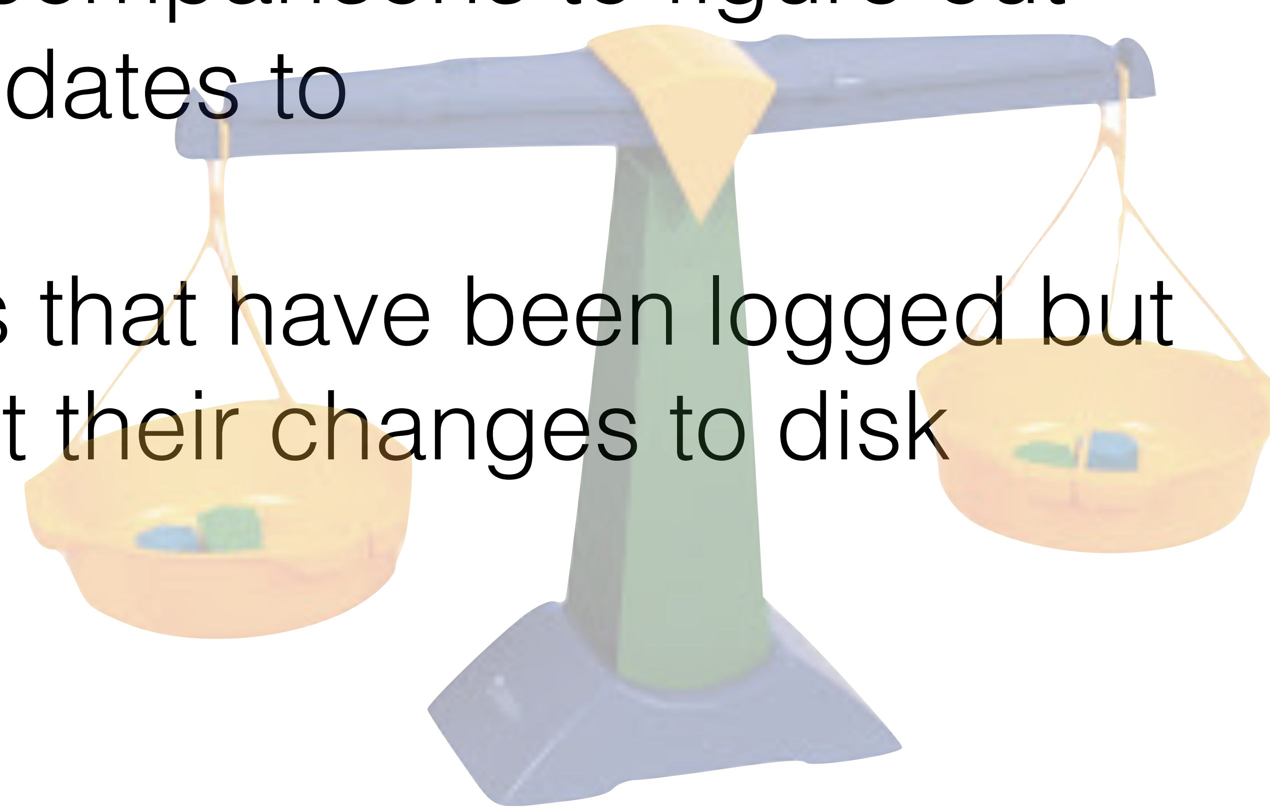


ARIES: Redo

- **Goal**: Repeat history to **reconstruct database state that was present during crash time**

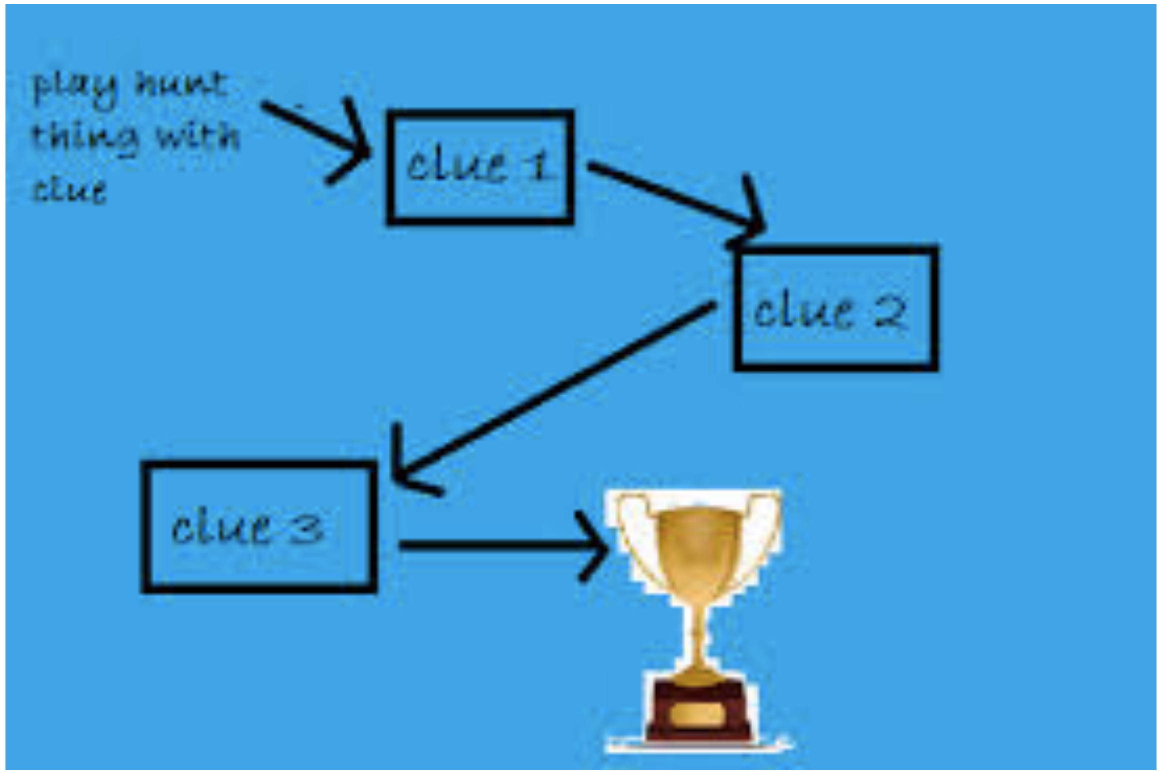
- Uses LSN and recLSN comparisons to figure out which pages to redo updates to

  - Reapplies all updates that have been logged but did not manage to get their changes to disk before crash



ARIES: Undo

- **Goal**: Undo effects of failed transactions
- **Loser transactions**: transactions active at crash
- Need to undo all loser transactions in reverse order
- Follow the lastLSN of each loser transaction until all effects of the transaction have been undone



Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART

Analysis and Redo phase get us here

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART

Undo phase

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60

UndoNext LSN = 20

Undo phase

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80, 85	CLR: Undo T3 LSN 50 T3 End

UndoNext LSN = 20

Undo phase

Crash Recovery Example

LSN	Log
00, 05	begin_checkpoint, end_checkpoint
10	update: T1 writes P5
20	update: T2 writes P3
30	T1 abort
40, 45	CLR: Undo T1 LSN 10 T1 End
50	update: T3 writes P1
60	update: T2 writes P5
	CRASH, RESTART
70	CLR: Undo T2 LSN 60
80, 85	CLR: Undo T3 LSN 50 T3 End
90	CLR: Undo T2 LSN 20 T2 End

UndoNext LSN = 20

Undo phase

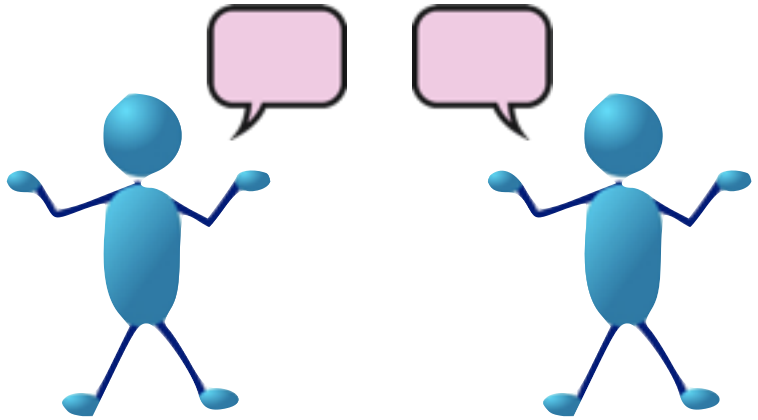
Selective Redo

- Redo phases are different from system to system (e.g., System R and DB2 use different redo phases)
- Selective redo is the idea of only redoing non-loser transactions
  - Can lead to trouble because we must log undos (for media recovery), but then we would attempt to redo the undo

Discussion Question

- If you are designing a system for transaction processing:
  - Would you redo “loser” transactions?
  - Would you use selective redo?
  - Would you do a checkpoint after the analysis phase?

Why or why not?



Summary

- ARIES is a database recovery algorithm
- Uses logs to identify what has happened
- Three phases: analysis, redo, and undo
- Different systems have different redo phases