# *Crash Recovery*

CMPSCI 445

Gerome Miklau

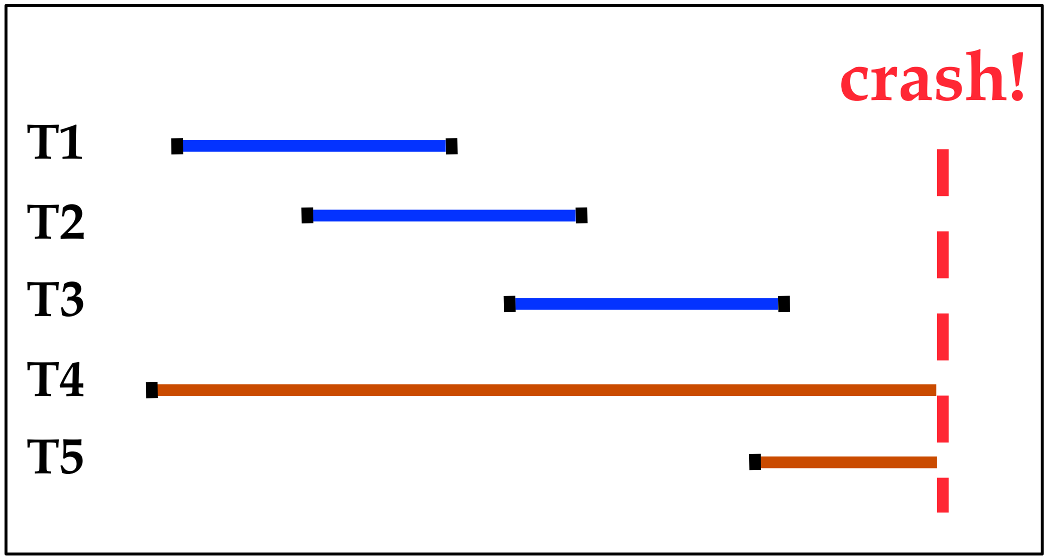
# *Review: the ACID Properties*

- Database systems ensure the ACID properties:
  - Atomicity: all operations of transaction reflected properly in database, or none are.
  - Consistency: each transaction in isolation keeps the database in a consistent state (this is the responsibility of the user).
  - Isolation: should be able to understand what's going on by considering each separate transaction independently.
  - Durability: updates stay in the DBMS!!!

# *Types of failure*

- Transaction failure
  - partially-executed transaction cannot commit
  - ➡ changes must be removed: ROLLBACK
- System failure
  - volatile memory lost
  - ➡ updates of committed Xact persist
  - ➡ updates of aborted or partial Xacts removed
- Media failure
  - corrupted storage media
  - ➡ database brought up-to-date using backup

# *Motivation*




- Desired Behavior after system restarts:
  - T1, T2 & T3 should be **durable**.
  - T4 & T5 should be **aborted** (effects not seen).

# *Undo and Redo*

- UNDO:
  - removing effects of incomplete or aborted transaction (for atomicity)
- REDO:
  - re-instating effects of committed transactions (for durability)
- The work the recovery subsystem must do to support UNDO and REDO depends on **key policies** of the buffer manager.

# *More on Steal and Force*

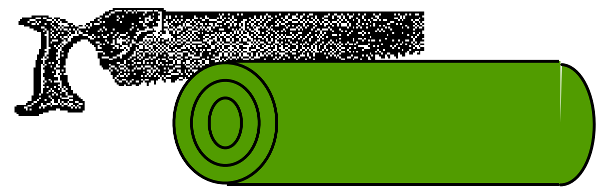
- **STEAL** (why enforcing Atomicity is hard)
  - *To steal frame F:* Current page in F (say P) is written to disk; some Xact holds lock on P.
    - What if the Xact with the lock on P aborts?
    - Must remember the old value of P at steal time (to support UNDOing the write to page P).
- **NO FORCE** (why enforcing Durability is hard)
  - What if system crashes before a modified page is written to disk?
  - Write as little as possible, in a convenient place, at commit time,to support REDOing modifications.

# *Handling the Buffer Pool*

- Force every write to disk?
  - Poor response time.
  - But provides durability.
- Steal buffer-pool frames from uncommited Xacts?
  - If not, poor throughput.
  - If so, how can we ensure atomicity?

| | No Steal | Steal |
|---|---|---|
| Force | Recovery **Trivial** | |
| No Force | | **Desired** |

# *Basic Idea: Logging*



- Record REDO and UNDO information, for every update, in a *log*.
  - Sequential writes to log (put it on a separate disk).
  - Minimal info (diff) written to log, so multiple updates fit in a single log page.
- Log: An ordered list of REDO/UNDO actions
  - Log record contains:

    Before image (for UNDO), After image (for REDO)
  - and additional control info (which we'll see soon).

<XID, pageID, offset, length, old data, new data>

# *Write-Ahead Logging (WAL)*

- The Write-Ahead Logging Protocol:
  - ① Must force the log record for an update *before* the corresponding data page is overwritten on disk.
  - ② Must write all log records for a Xact *before commit*.
- #1 guarantees Atomicity.
- #2 guarantees Durability.

- Exactly how is logging and recovery done?
  - the ARIES algorithm (we won't see details of this in this class)

# *Log Records*

**LogRecord fields:**

prevLSN
XID
type
pageID
length
offset
before-image
after-image

**update** records only (pageID, length, offset, before-image, after-image)

Possible log record types:

- **Update**
- **Commit**
- **Abort**
- **End** (signifies end of commit or abort)
- Compensation Log Records (CLRs)
  - for UNDO actions

# *The Big Picture: What's Stored Where*

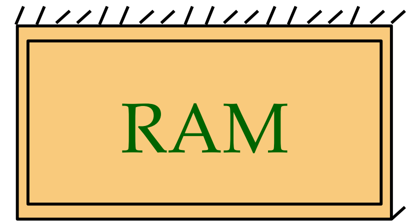

**LogRecords**

prevLSN
XID
type
pageID
length
offset
before-image
after-image


**Data pages**

each
with a
pageLSN

**master record**




**Xact Table**

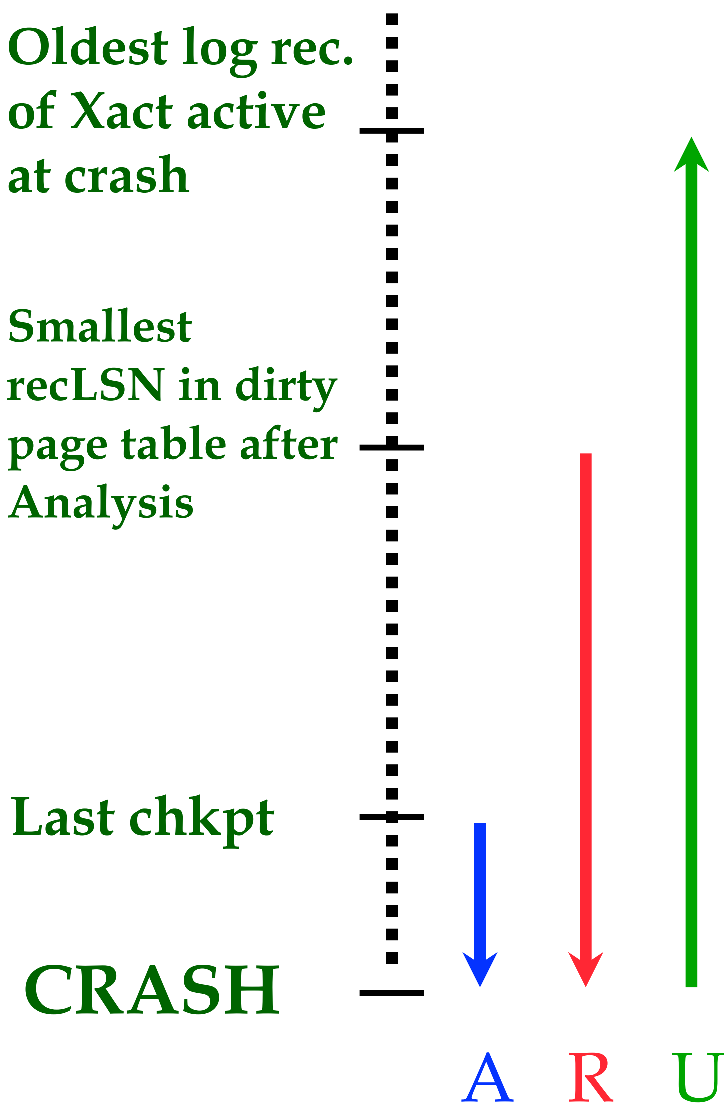
lastLSN
status

**Dirty Page Table**

recLSN

**flushedLSN**

# *Crash Recovery: Big Picture*




- Start from a checkpoint (found via master record).
- Three phases. Need to:
  - Analysis: Figure out which Xacts committed since checkpoint, which failed.
  - REDO ***all*** actions.
    - (repeat history)
  - UNDO effects of failed Xacts.

# *Crash during recovery*

- Crashes during UNDO handled by logging CLRs
- What happens if system crashes during Analysis or Redo?
  - Analysis: all work is lost, but analysis begins again.
  - Redo: Just redo again -- redo idempotent. Some pages may have been written to disk before crash but this will be evident.

# *Summary of Logging/Recovery*

- Recovery Manager guarantees Atomicity & Durability.
- Use WAL to allow STEAL/NO-FORCE w/o sacrificing correctness.