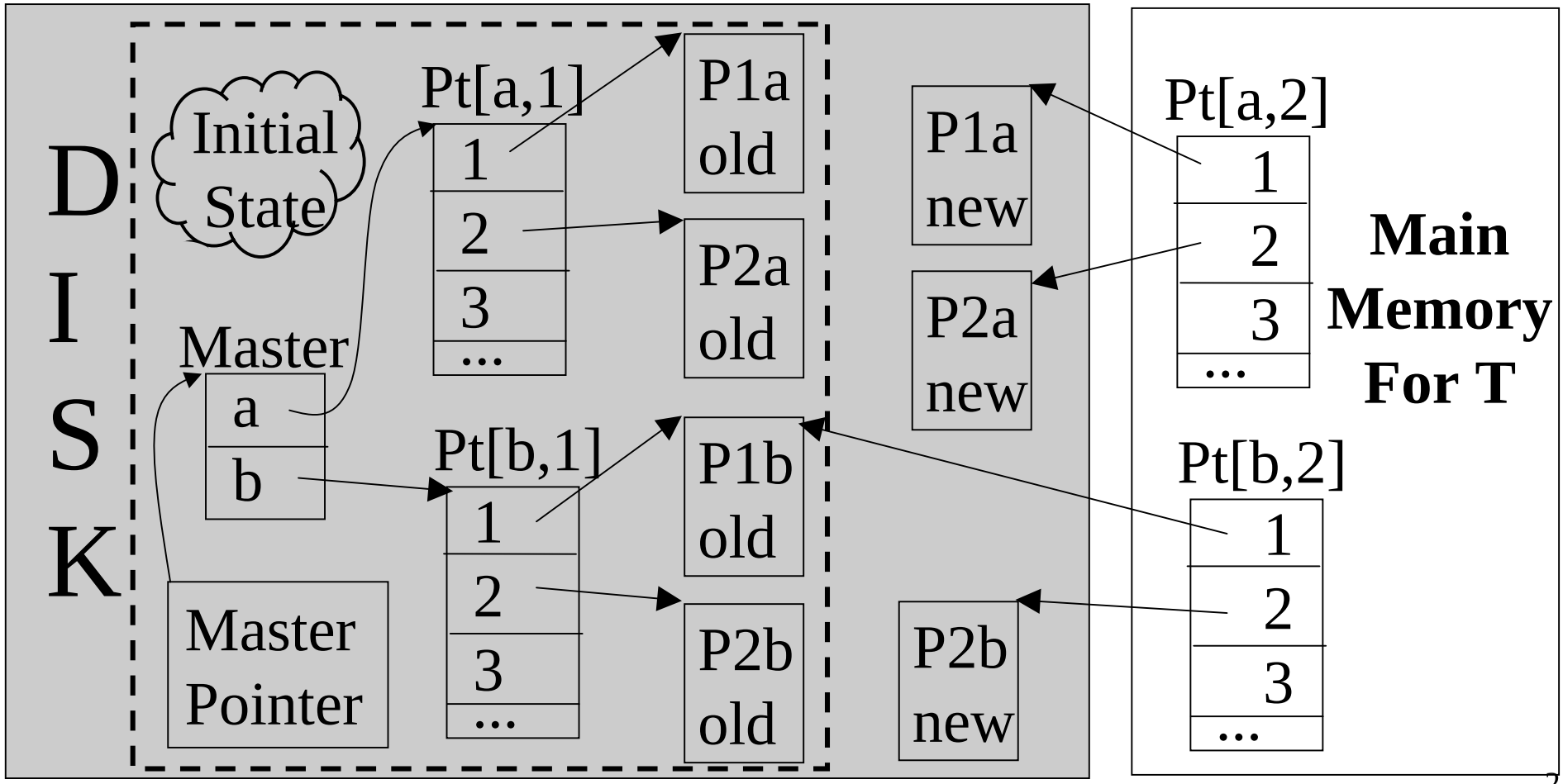
Shadow Paging

CSE593 Transaction Processing

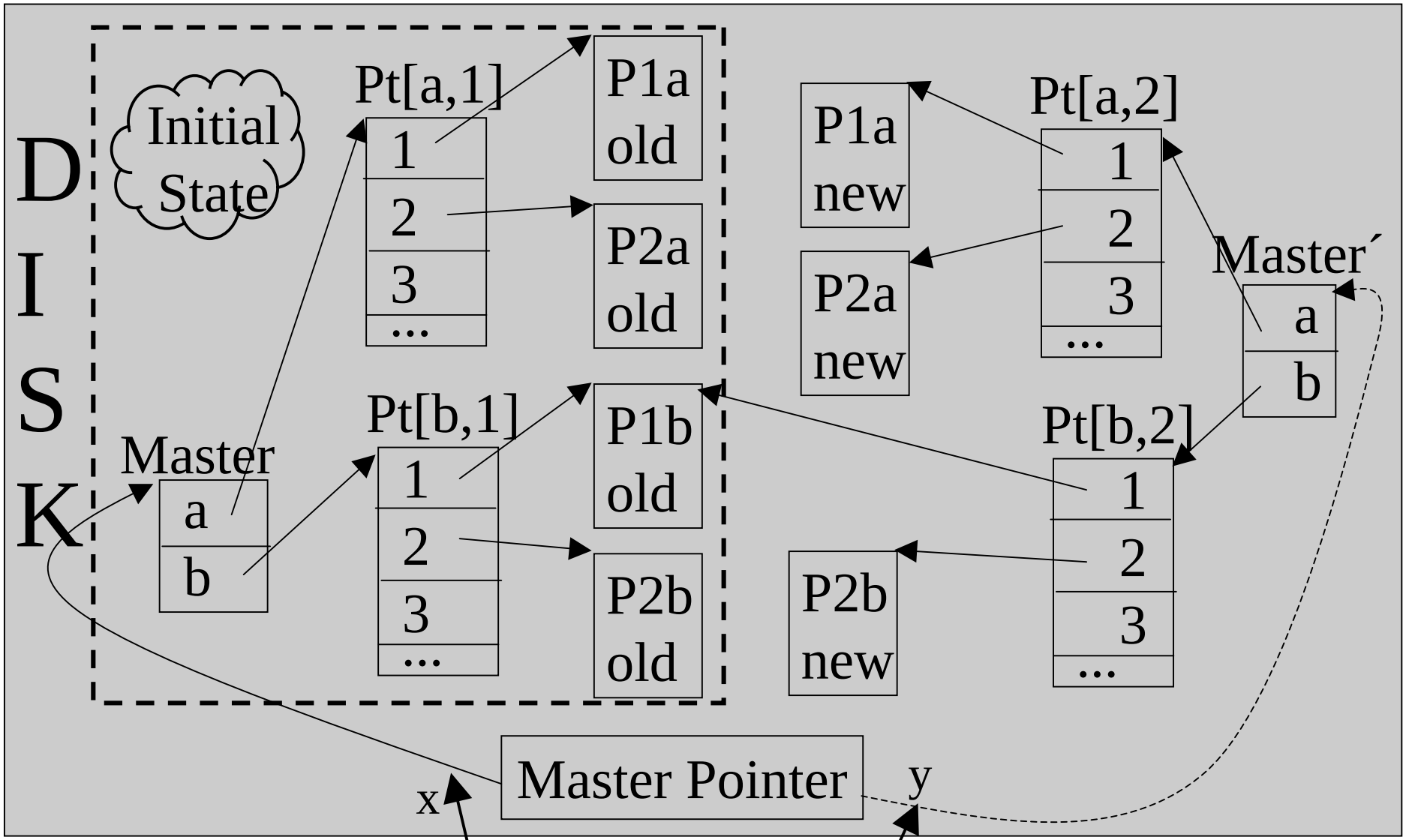
Philip A. Bernstein

Shadow Paging

- Each file is managed via a page table P
  - Each transaction T updates the file via a private page table
  - Commit T by replacing the public page table by a private one
  - Example: suppose DB has two files, “a” and “b”



1. To commit, first copy Pt[a,2], Pt[b,2], and Master´ to disk



2. Then replace this pointer (x) by this one (y)

Shadow Paging with Shared Files

- What if two transactions update different pages of a file?
  - If they share their main memory copy of the page table, then committing one will commit the other's updates too!
- One solution: File-grained locking
  - Poor concurrency
- Better solution: use a private copy of page table, per transaction. To commit T, *within a critical section*:
  - get a private copy of the last committed value of the page table of each file modified by T
  - update their entries for pages modified by T
  - store the updated page tables on disk
  - write a new master record and master pointer, thereby installing the update *just for T* (// end of critical section)

Shadow Paging in Practice

- Reference: R. Lorie, “Physical Integrity in a Large Segmented Database”*ACM Trans. on DB Sys*., March 1977.
- Used in the Gemstone OO DBMS.
- Not good for TPC
  - count disk updates per transaction
  - how to do record level locking?