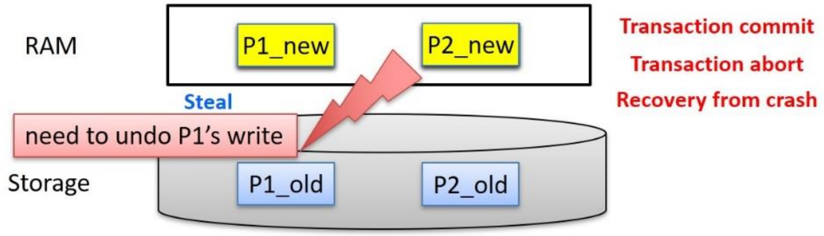
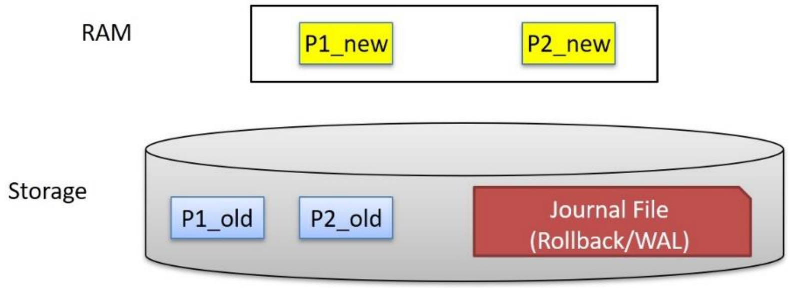
SQLite: Transactional Atomicity

- For example, two pages (P1,P2) are updated by a transaction
- Transactional atomicity is all or nothing
  - Force policy need to write both pages at commit (**ALL**)
  - Steal policy allows to overwrite P1 prior to commit so that we need to undo P1's write upon abort (**NOTHING**)
  - Recovery from crash checks whether both pages are successfully written, and if not, need to undo (**ALL or NOTHING**)



- Two journal modes
  - Rollback journal (RBJ, default) and Write Ahead Logging(WAL)
- Why SQLite's own journaling modes, instead of file system journaling?
  - Portability : every file system does not support journaling
  - Steal policy semantics


RAM
P1_new
P2_new
Storage
P1_old
P2_old
Journal File
(Rollback/WAL)