Conor Reisman

5/23/2013

CSE444

Lab 5

Design Decisions

I decided to implement my rollback starting from the end of the transaction and move up the log undoing every update record belonging to the loser transaction. This is not the most efficient method as a lot of seeking is involved. Also, I may do more writing than is necessary since I only need to write the earliest version of a page and not all versions. However, it was much simpler to implement.

In my redo phase, when I encounter an abort, I immediately rollback that transaction even though with a more thorough analysis I could determine whether I actually need to redo it in the first place, or undo it at all.

API Changes

I modified the invariant of the before image for a page to mean the data that is currently on disk. This way the image changes with each flush instead of with each transaction. It was easier to implement, even though it's higher overhead.

Incomplete/Missing Implementation

Nothing that was required for lab 5 was left unimplemented.

Other

A possible way to break the invariant on the before image based on the spec.

BufferPool is initialized with room for one page.
BufferPool reads page 0 of file 0 into memory. It is initially empty with a blank before image.
A tuple is inserted into this page. Page(0, 0) now contains the tuple (1, "foo").
The pages are flushed to disk.
BufferPool reads page(0, 1), which requires evicting page(0, 0).
BufferPool reads page(0, 0), which requires evicting page(0, 1). The before image is now a page containing the tuple (1, "foo"), since that's what it read from disk.

This contradicts what the spec implies should happen, since it implies the before image should always be the most recently committed version and not the most recently written to disk.