Project5

Transaction Logging & Recovery

Log Manager

- Your database systems don’t support transaction yet.
- Implement transaction API that can support ‘Atomicity’ and ‘Durability’ using your own log manager.
- Your log manager should satisfy those properties.
  - No force (REDO) & Steal (UNDO) policy
  - Write Ahead Logging (WAL)
  - Recovery when initializing DB

Project Specification

- Your library (libbpt.a) should provide those API services.
- Transaction APIs
  - **int begin_transaction();**
    - Allocate transaction structure and initialize it.
    - Return 0 if success, otherwise return non-zero value.
  - **int commit_transaction();**
    - Return 0 if success, otherwise return non-zero value.
    - User can get response once all modification of transaction are flushed to a log file.
    - If user get successful return, that means your database can recover committed transaction after system crash.
  - **int abort_transaction();**
    - Return 0 if success, otherwise return non-zero value.
    - All affected modification should be canceled and return to old state.

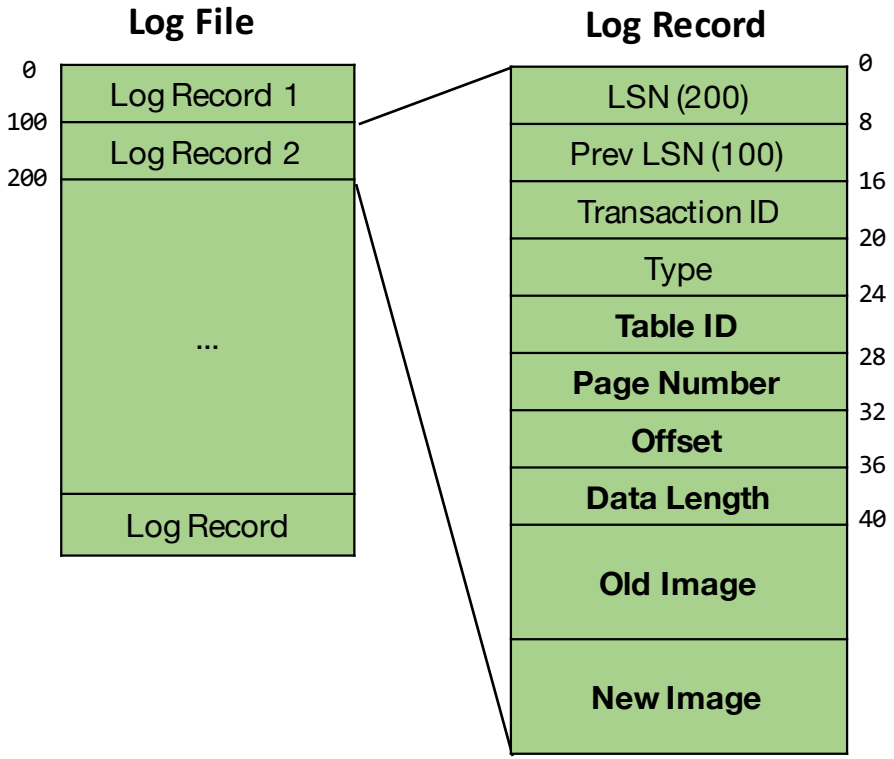
Project Specification

➢ Your library (libbpt.a) should provide those API services.

1. **int init_db (int buf_num);**
   - Recovery before database initialization.
2. **int open_table (char * pathname);**
   - **We limit the file name format as "DATA[NUM]" (For example, there should be data files named like "DATA1", "DATA2", …)**
   - **Return value that indicates the table id should be NUM. (That means, data file whose file name is "DATA3" has its table id as 3 from now on).**
3. **int insert (int table_id, int64_t key, char * value);**
4. **char * find (int table_id, int64_t key);**
5. **int delete (int table_id, int64_t key);**
6. **int update(int table_id, int64_t key, char* value);**
   - Find the matching key and modify the value, where value size $\leq$ 120 Bytes.
   - Return 0 if success, otherwise return non-zero value.
7. **int close_table(int table_id);**
8. **int shutdown_db(void);**

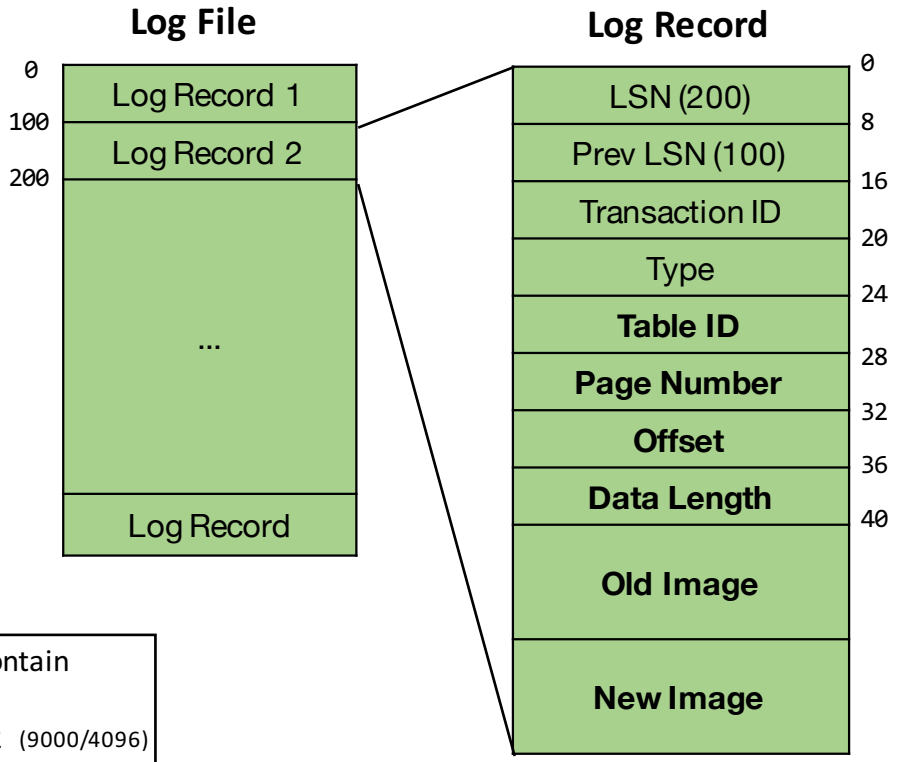
Log File

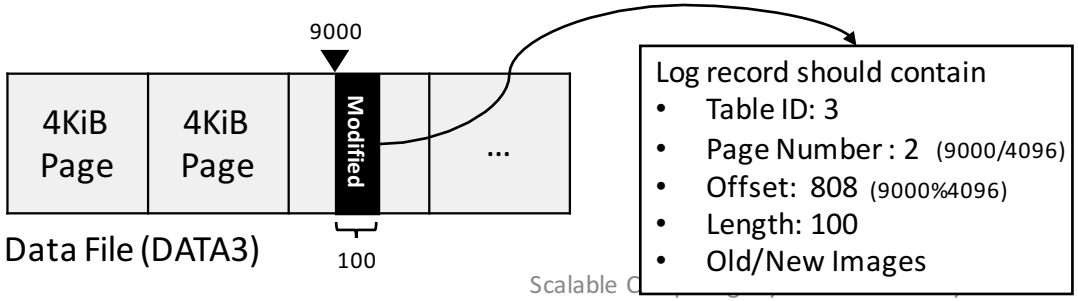
- Log file is a sequence of log records.
- Log record is consisted of
  - LSN: End offset of a current log record.
  - Prev LSN: LSN of the previous log record.
  - Transaction ID: Indicates the transaction that triggers current log record.
  - Type: The type of current log record.
    - BEGIN(0)
    - UPDATE (1)
    - COMMIT (2)
    - ABORT (3)



Log File

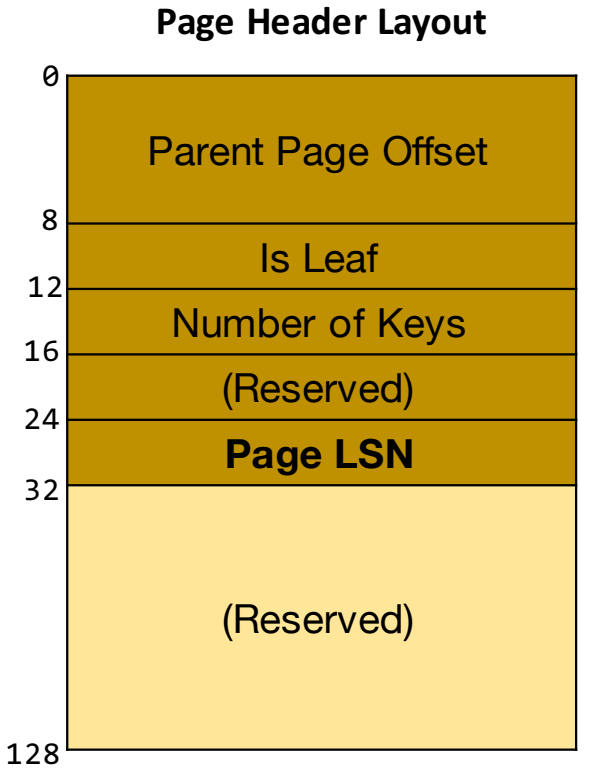
- Log file is a sequence of log records.
- Log record is consisted of
  - Table ID: Indicates the data file. (data file name should be like “DATA[Table ID]”)
  - Page Number: Page that contains the modified area.
  - Offset: Start offset of the modified area within a page.
  - Data Length: The length of modified area.
  - Old Image: Old contents of the modified area.
  - New Image: New contents of the modified area.



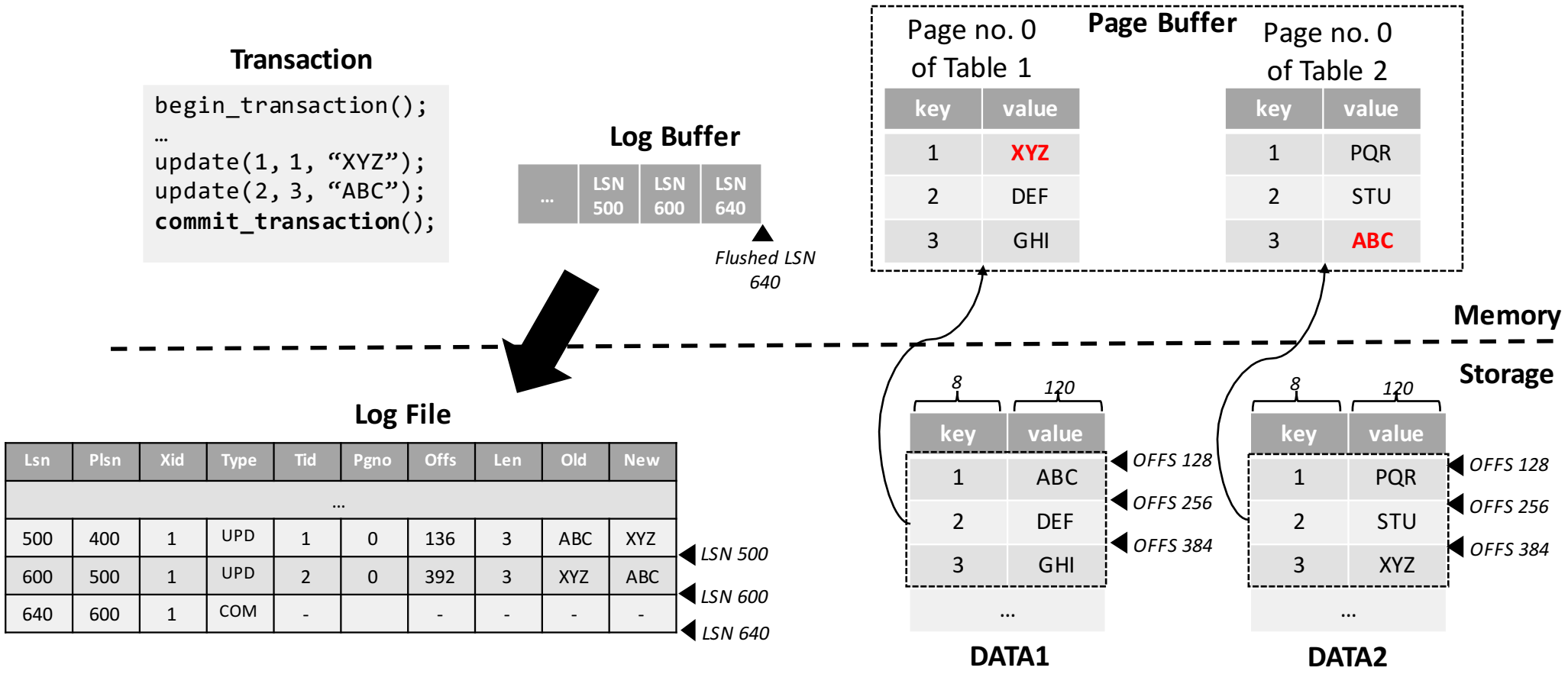


Page Header Layout

- We should maintain a page LSN information from every on-disk image.
- Page LSN indicates the last updated version of this on-disk page.
- Maintain a page LSN value (8 bytes) located at the page header structure starting from byte offset 24.
- Every page including header page should maintain that field.



Project Example



HYU 한양대학교 HANYANG UNIVERSITY

Project Specification

- You should implement ARIES based recovery that you learned from the lecture but,
  - You don’t have to implement strict 2PL or other concurrency control since we assume that only single transaction would be working at given time.
  - Also, we don’t have to consider double write since we assume that torn page write would not occur.
  - Checkpoint is not considered from this project.
- We will check the correctness of recovery by executing simple transactions and triggering system crash like below.

```
begin_transaction();
update(1, 3, “XYZ”);
commit_transaction();
begin_transaction();
update(1, 2, “XXX”);
exit() // system crash
```

Submission

- Please upload your implemented source directory to GitLab with this format.
  - Base directory name should be “project_final”
  - Base directory should have your ‘Makefile’.
  - Compile should be done by typing ‘make’ command from your base directory. (NOTE that, we will not grade your code if compile is failed.)
  - After compile, output library (libbpt.a) should be located in base directory or any sub-directories in “project_final”.