Project3

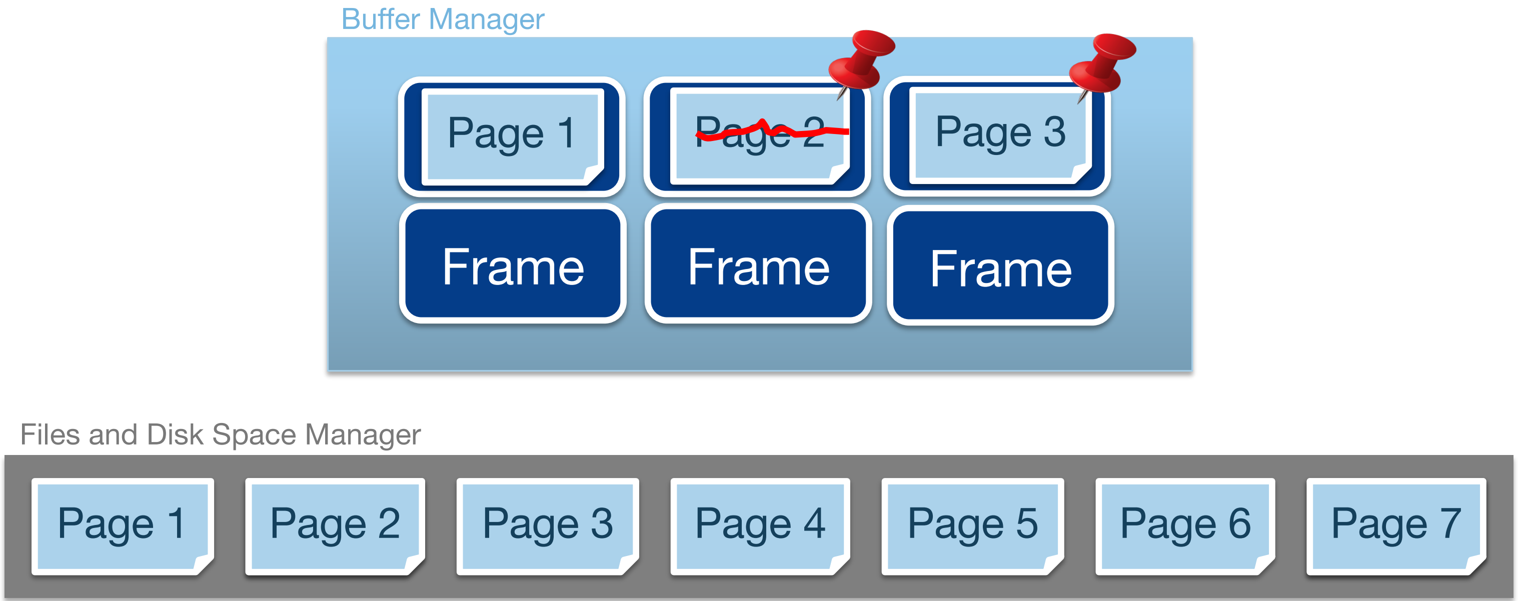
Buffer management

Project Hierarchy

- Your project hierarchy should be like this.
  - Your_hconnect_repo
    - project3
      - include/
      - lib/
      - Makefile
      - src/
- If your Makefile doesn't make libbpt.a library file at the exact path, you'll get zero score. (your_hconnect_repo/project3/lib/libbpt.a)

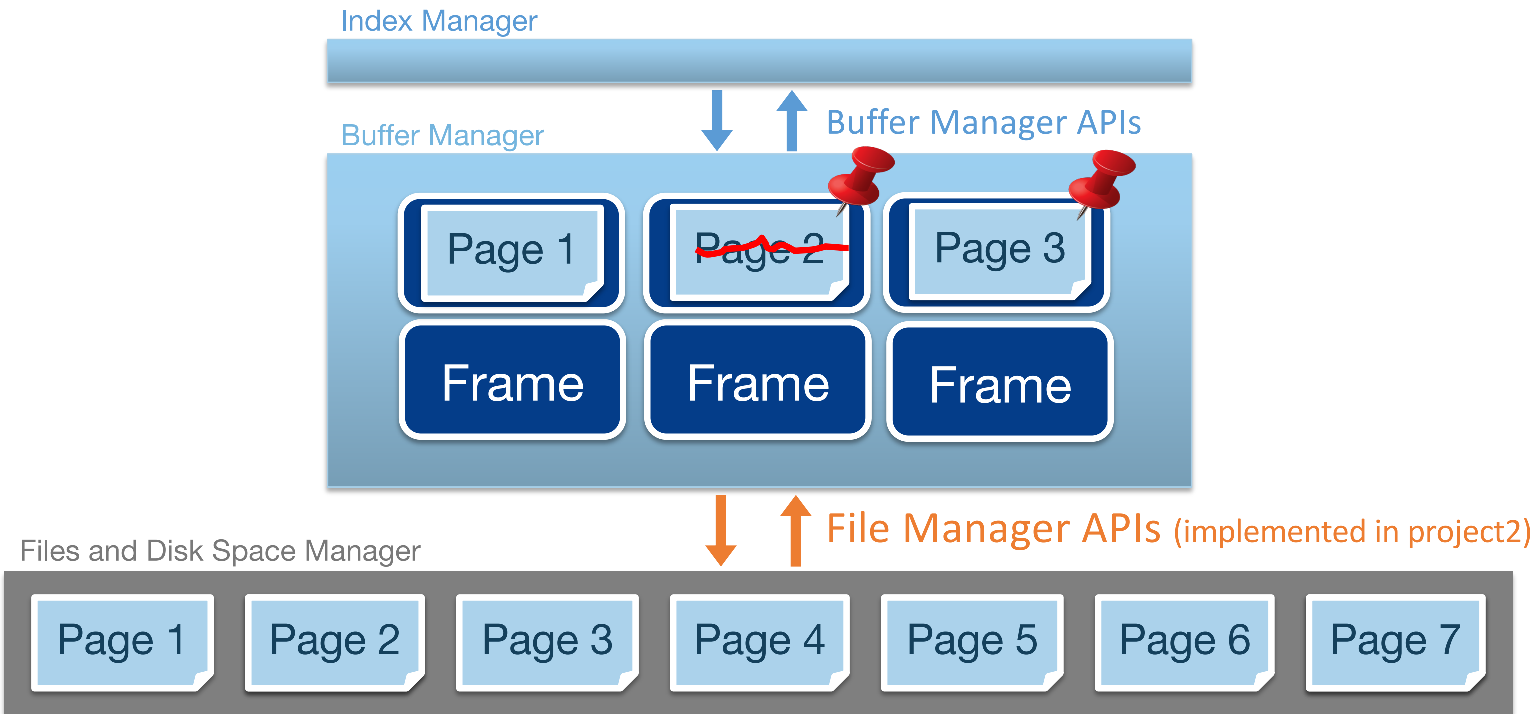
Buffer Management

- Current disk-based b+tree doesn't support buffer management.
- Our goal is to implement **in-memory buffer manager** to caching on-disk pages.



Buffer Management Layer

- File manager APIs should be called only within the buffer manager layer.



Project Specification

- Define the buffer block structure, which must contain at least those fields.
  - **Physical frame**: containing up to date contents of target page.
  - **Table id:** the unique id of table (per file)
  - **Page number**: the target page number within a file.
  - **Is dirty:** whether this buffer block is dirty or not.
  - **Is pinned:** whether this buffer block is accessed right now.
  - **LRU list next (prev)** : buffer blocks are managed by LRU list.
  - Other information can be added with your own buffer manager design.

Buffer Structure

frame (page size : 4096 bytes)
table_id
page_num
is_dirty
is_pinned
next/prev of LRU

⋮

Project Specification

- Implement database initialization function.
  - **int init_db (int num_buf);**
  - Allocate the buffer pool (array) with the given number of entries.
  - Initialize other fields (state info, LRU info..) with your own design.
  - If success, return 0. Otherwise, return non-zero value.
- **open_table interface**
  - **int open_table (char *pathname);**
  - Open existing data file or create one if not existed. You must give the same table id when db opens the same table more than once after init_db(). (the length of pathname <= 20)
  - If success, return the **unique table id**, which represents the own table in this database. (Return negative value if error occurs)
  - You have to maintain a table id once open_table() is called, which is matching file descriptor or file pointer depending on your previous implementation. (table id ≥ 1 and maximum allocated id is set to 10)

Project Specification

- A table id needs to be passed to the index manager APIs to select the table where the operation will be executed.
  - **int db_insert (int table_id, int64_t key, char * value);**
  - **int db_find (int table_id, int64_t key, char* ret_val);**
  - **int db_delete (int table_id, int64_t key);**

Project Specification

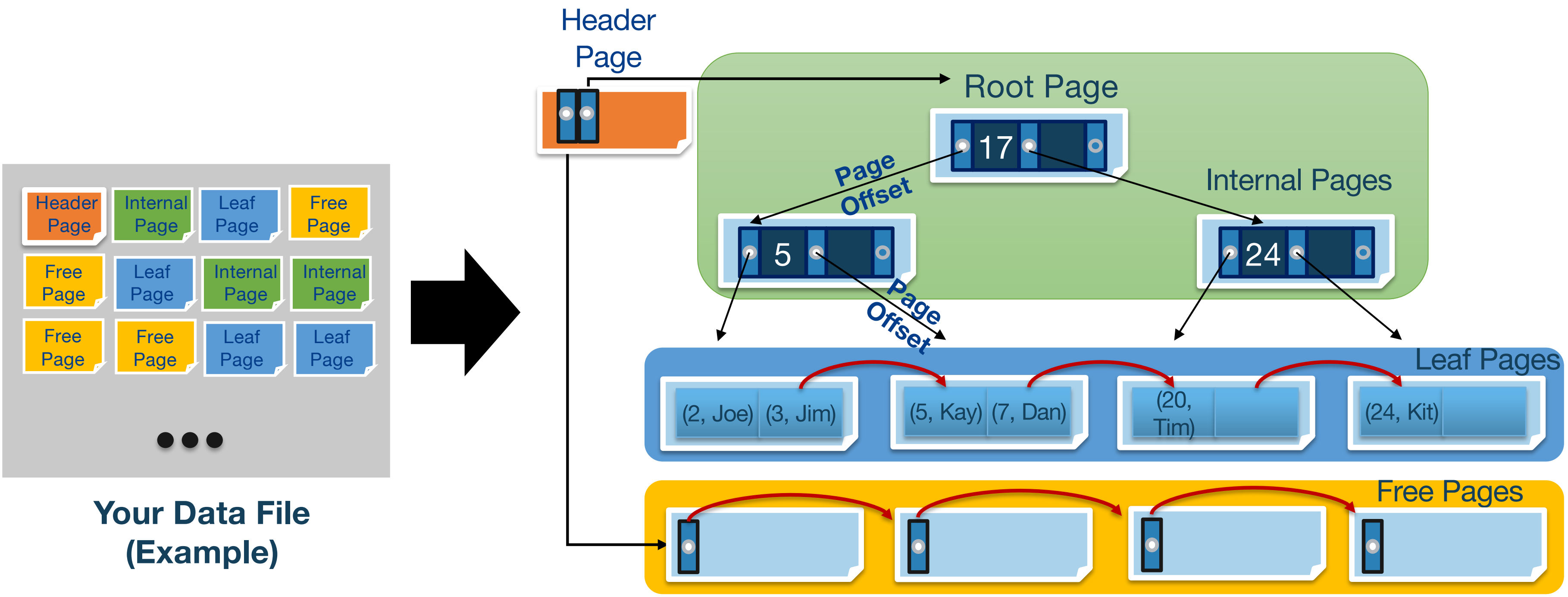
- Implement **close_table** interface.
  - **int close_table(int table_id);**
  - Write all pages of this table from buffer to disk.
  - If success, return 0. Otherwise, return non-zero value.
- Implement database shutdown function.
  - **int shutdown_db();**
  - Flush all data from buffer and destroy allocated buffer.
  - If success, return 0. Otherwise, return non-zero value.

Project Specification

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

1. **int init_db (int buf_num);**
   - Initialize buffer pool with given number and buffer manager.
2. **int open_table (char * pathname);**
   - Open existing data file using 'pathname' or create one if not existed. If success, return **table_id**.
3. **int db_insert (int table_id, int64_t key, char * value);**
4. **int db_find (int table_id, int64_t key, char* ret_val);**
5. **int db_delete (int table_id, int64_t key);**
6. **int close_table(int table_id);**
   - Write the pages relating to this table to disk and close the table.
7. **int shutdown_db(void);**
   - Destroy buffer manager.

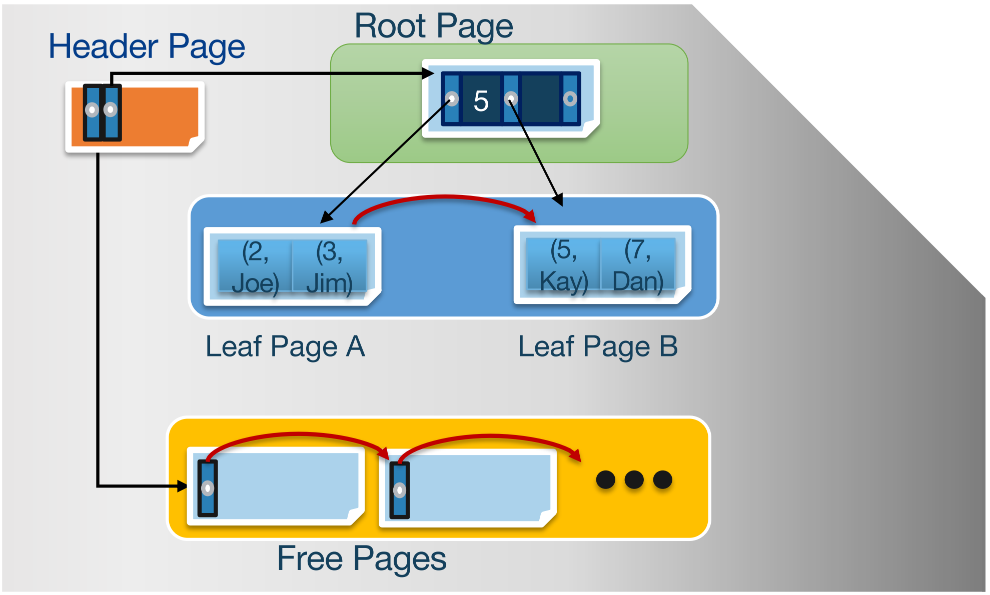
So far..

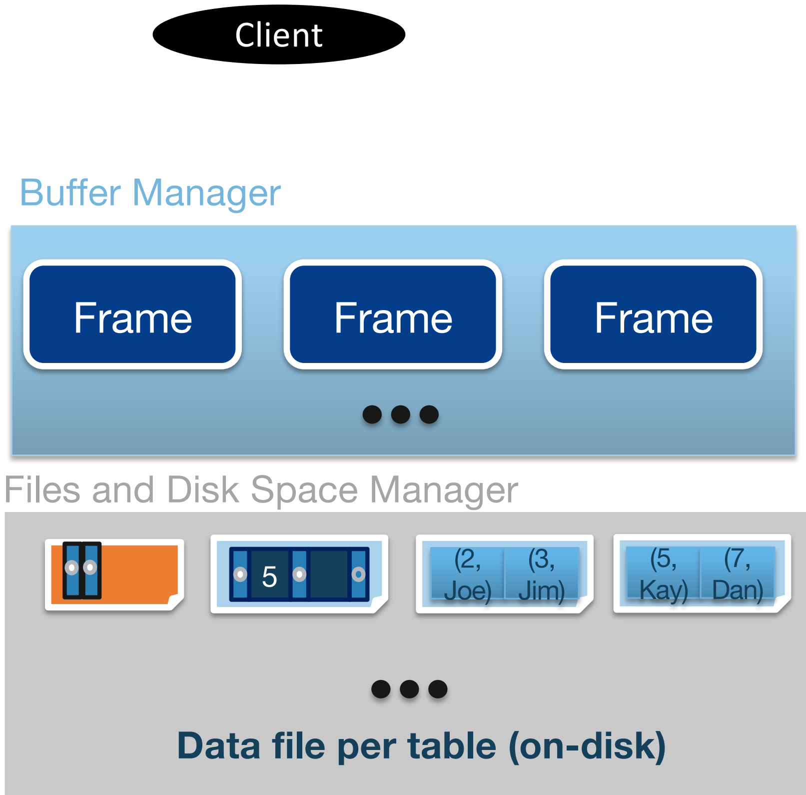


Header Page
Internal Page
Leaf Page
Free Page
Free Page
Leaf Page
Internal Page
Internal Page
Free Page
Free Page
Leaf Page
Leaf Page
Your Data File (Example)
Header Page
Root Page
17
Page Offset
Internal Pages
5
24
Page Offset
Leaf Pages
(2, Joe) (3, Jim)
(5, Kay) (7, Dan)
(20, Tim)
(24, Kit)
Free Pages

Buffer Management

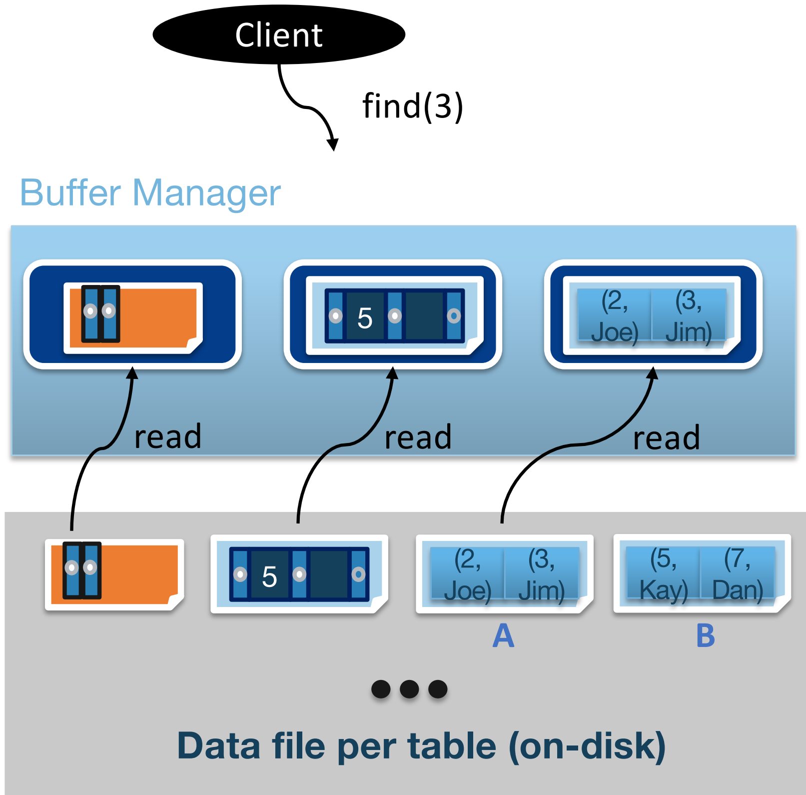
- Assume the on-disk pages are stored like below form.





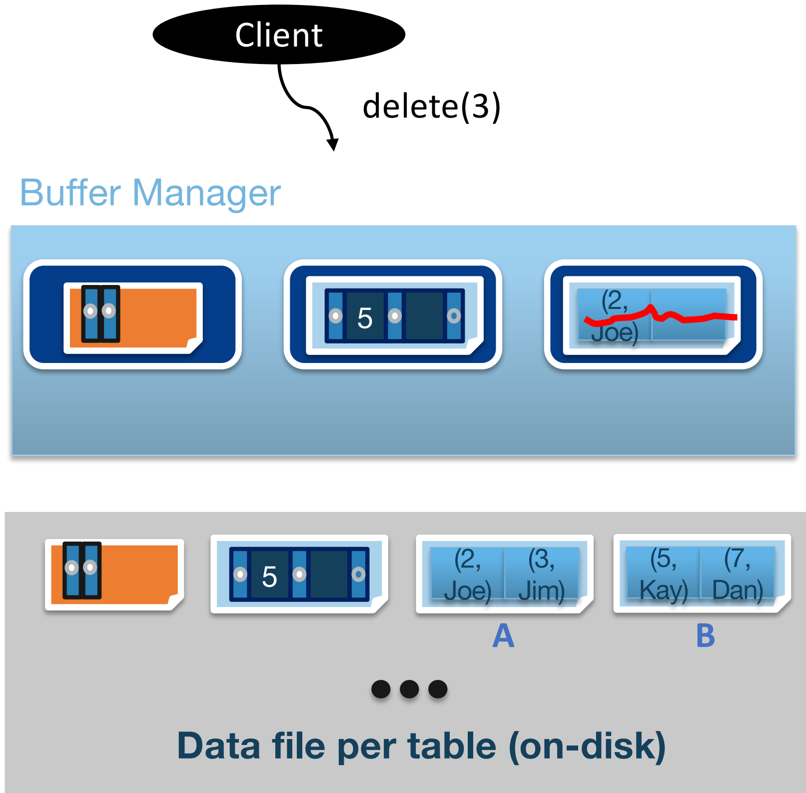
Buffer Management

- First, search the page from the buffer pool.
- If the page is not in the buffer pool (i.e, cache-miss occurs), read the page from disk and maintain this page in a buffer block.
- While indexing from the root to the leaf page A (where key 3 is located), the header page and the root page (internal page) are also read by the buffer manager.



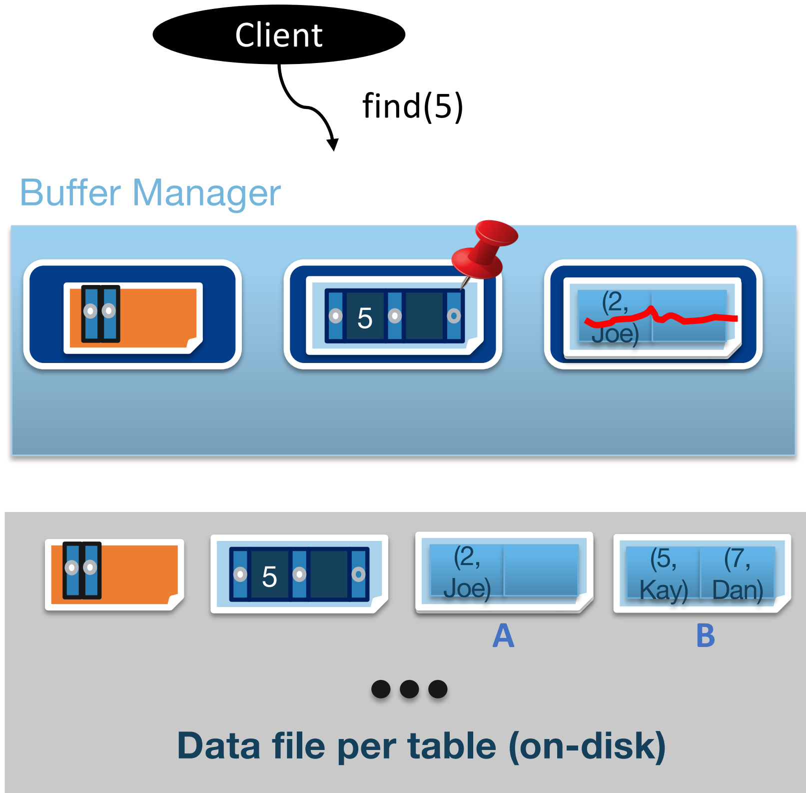
Buffer Management

- After reading the page to the buffer, update operation can be handled in the buffer (memory).
- So "delete key 3" operation occurs in the buffer, which makes that page marked to dirty.



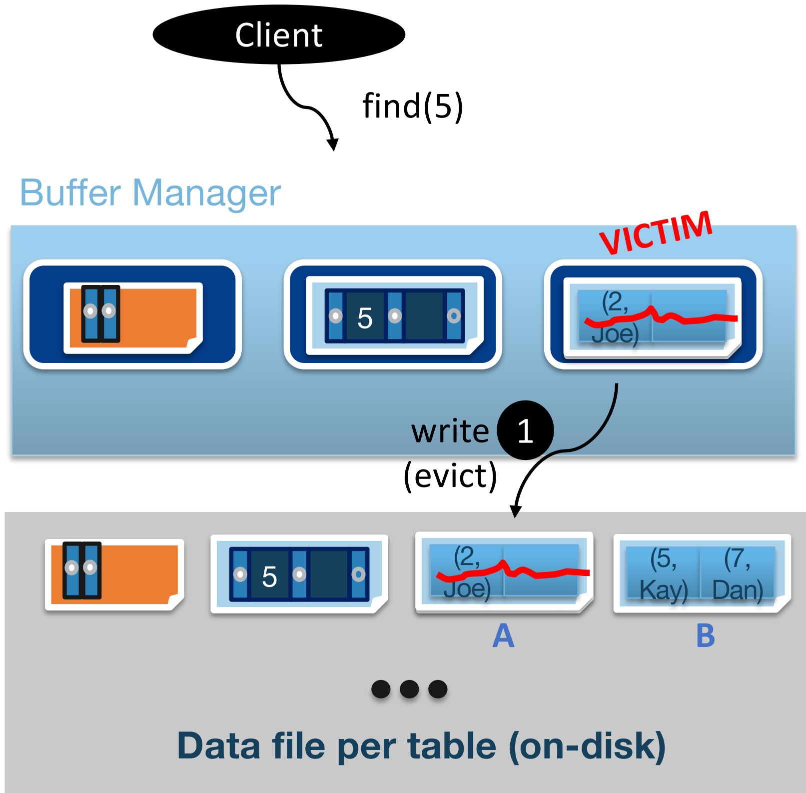
Buffer Management

- Dirty page is written to disk when those page is selected to the victim of LRU policy.
- Assuming the example shown right, find(5) tries to read root page.



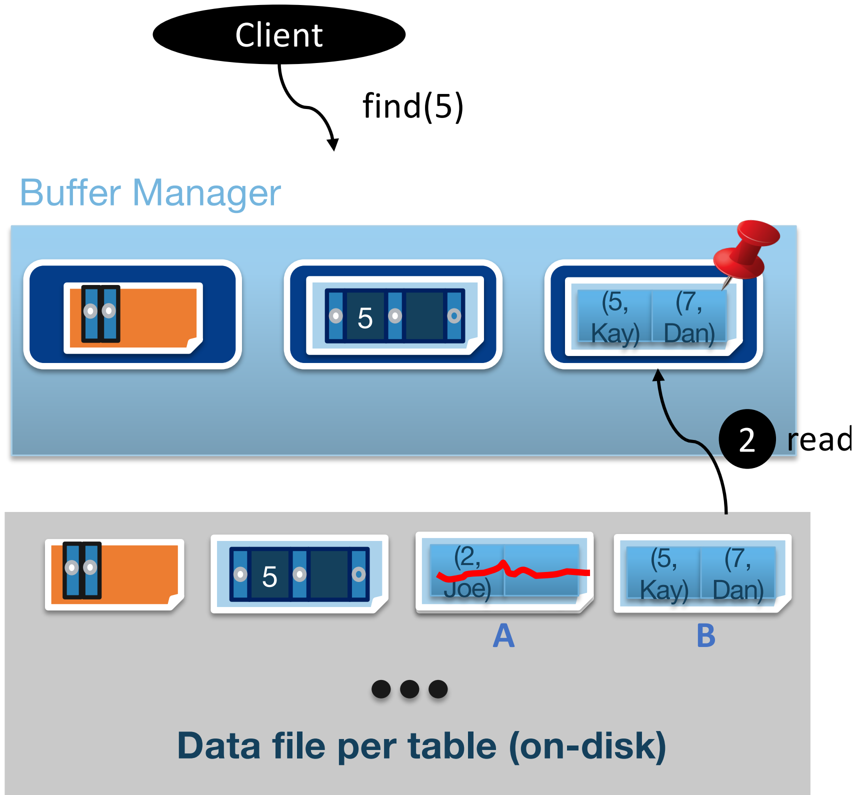
Buffer Management

- Dirty page is written to disk when the page is selected to the victim of LRU policy.
- Assuming the example shown right, find(5) tries to read the leaf page B which triggers page eviction. (pinned page should not be the victim of eviction.)
- If the victim page is marked as dirty, write data to disk first. 1



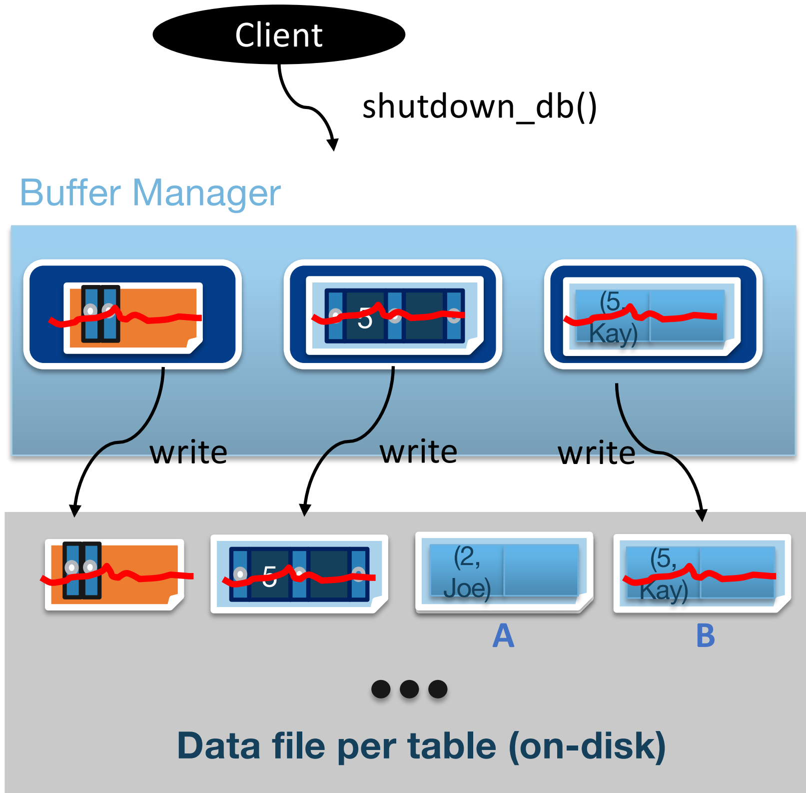
Buffer Management

- Dirty page is written to disk when those page is selected to the victim of LRU policy.
- Assuming the example shown right, find(5) tries to read the leaf page B which triggers page eviction. (pinned page should not be the victim of eviction.)
- If the victim page is marked as dirty, write data to disk first. 1
- Then read another page from disk. 2



Buffer Management

- close_table() or shutdown_db() writes out all dirty buffer block to disk.
- close_table() writes out the pages only from those relating to given table_id.
- This command can provide synchronous semantic (durability) to the user but lose performance.



File Manager APIs

- The File Manager APIs should be properly changed for Project3.

Submission

- Implement in-memory buffer manager and submit a report (Wiki) including your design
  - Deadline: Nov 1 11:59pm
- We'll only score your commit before the deadline and your submission after the deadline will not accepted

Thank you