Milestone 2: Single-threaded, Durable L-Store

ECS165A - WQ2021

Sean Carnahan, Reese Lam, Gabriel Vazquez,
Quang-Long Tran, Aly Kapasi

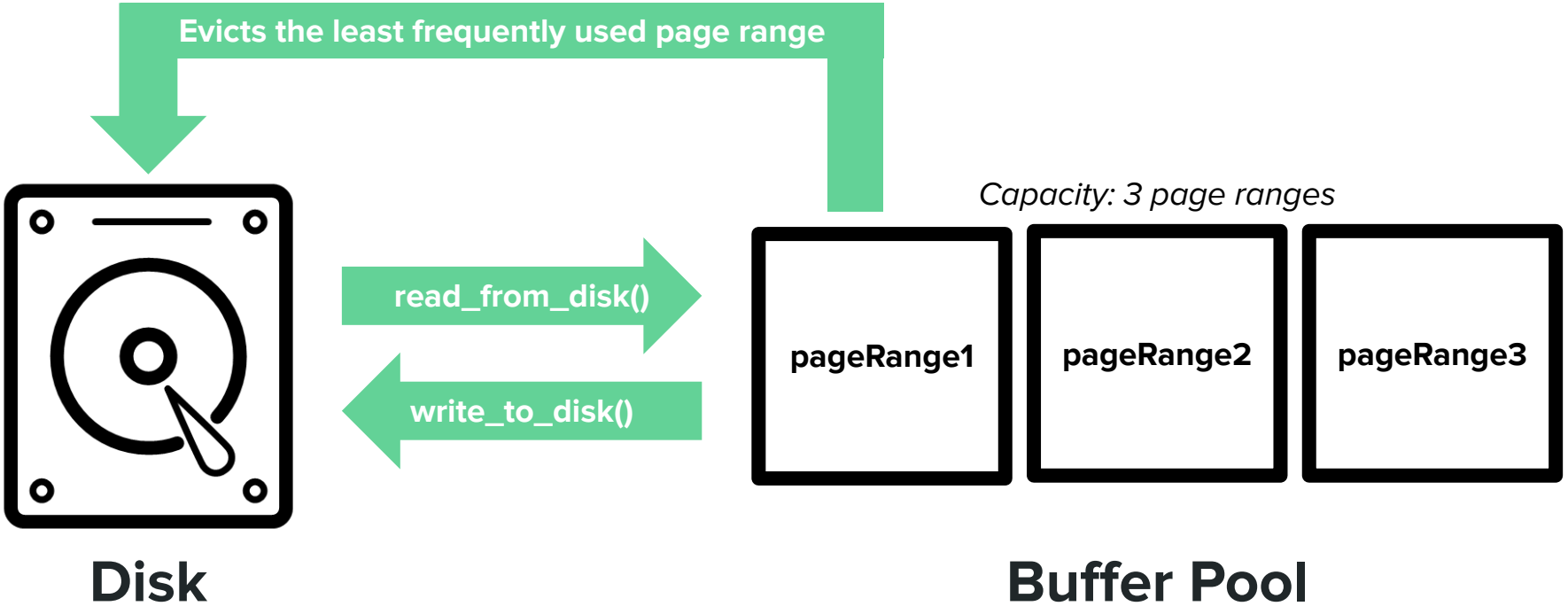
Agenda

Buffer Pool

Merge

Index

Buffer Pool: In-Memory Page Ranges

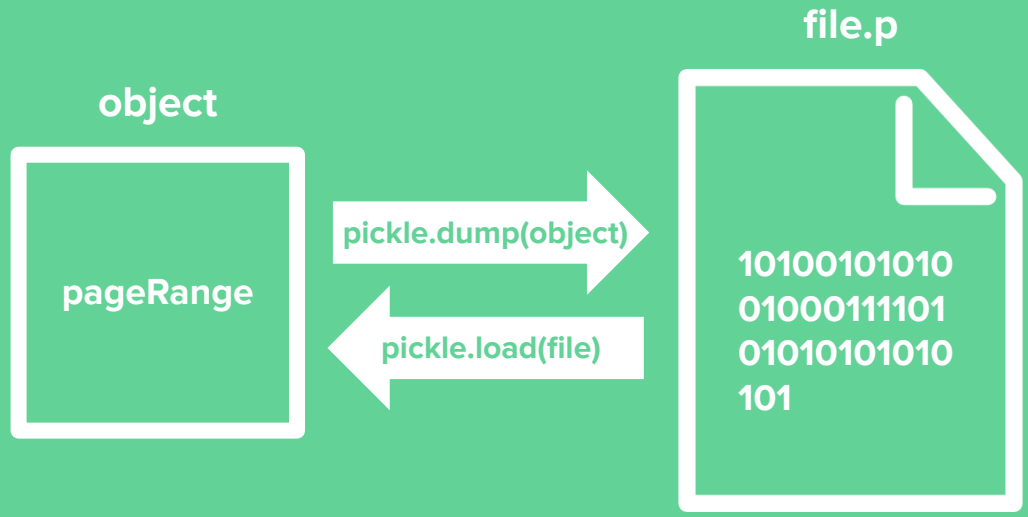


Buffer Pool: Pickling

The Pickle Python Library is used to serialize and deserialize python objects.

Pickling an object is the process of converting it into a bytestream. We can then store it within a file.

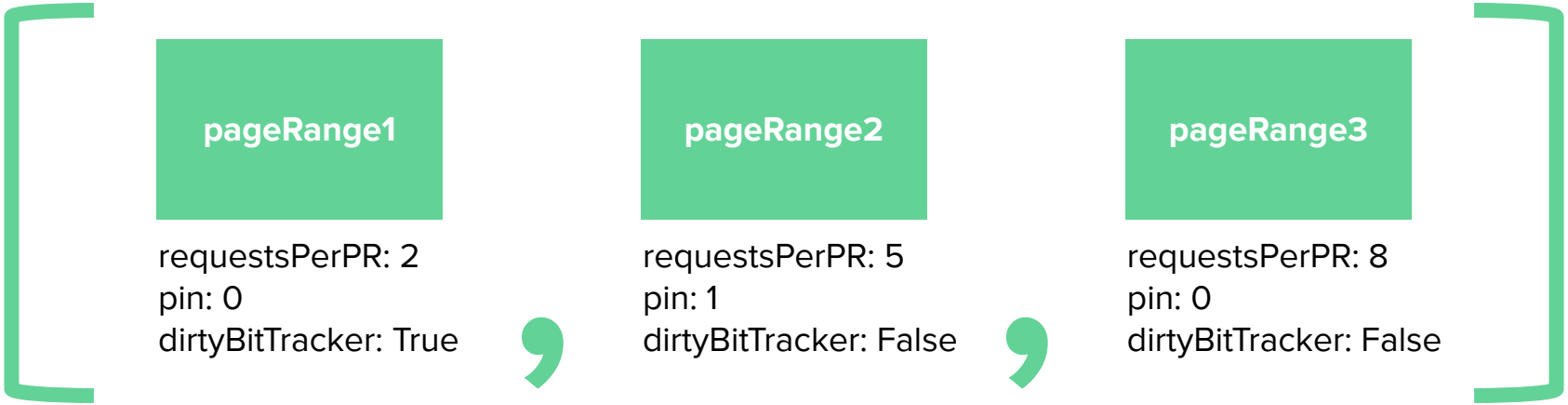
Unpickling an object converts the bytestream back into a python object.



Buffer Pool: LFU Eviction Policy

The bufferpool evicts the **Least Frequently Used** page range by:

1. Checking the number of requests which is quickly done through a sorted list
2. Skipping pinned page ranges; pin ≠ 0 signifies that they are in use
3. Writing the page range to disk if its dirty bit tracker equals true



Agenda

Merge: Thread Object & Buffer Pool

Merge Thread Object

- Created with database
- Starts when database is opened
- Checks if there are PageRanges from the buffer pool to perform a merge on
- Calls merge and resets the number of unmerged tail records for the PageRange merged
- Terminates when the database is closed

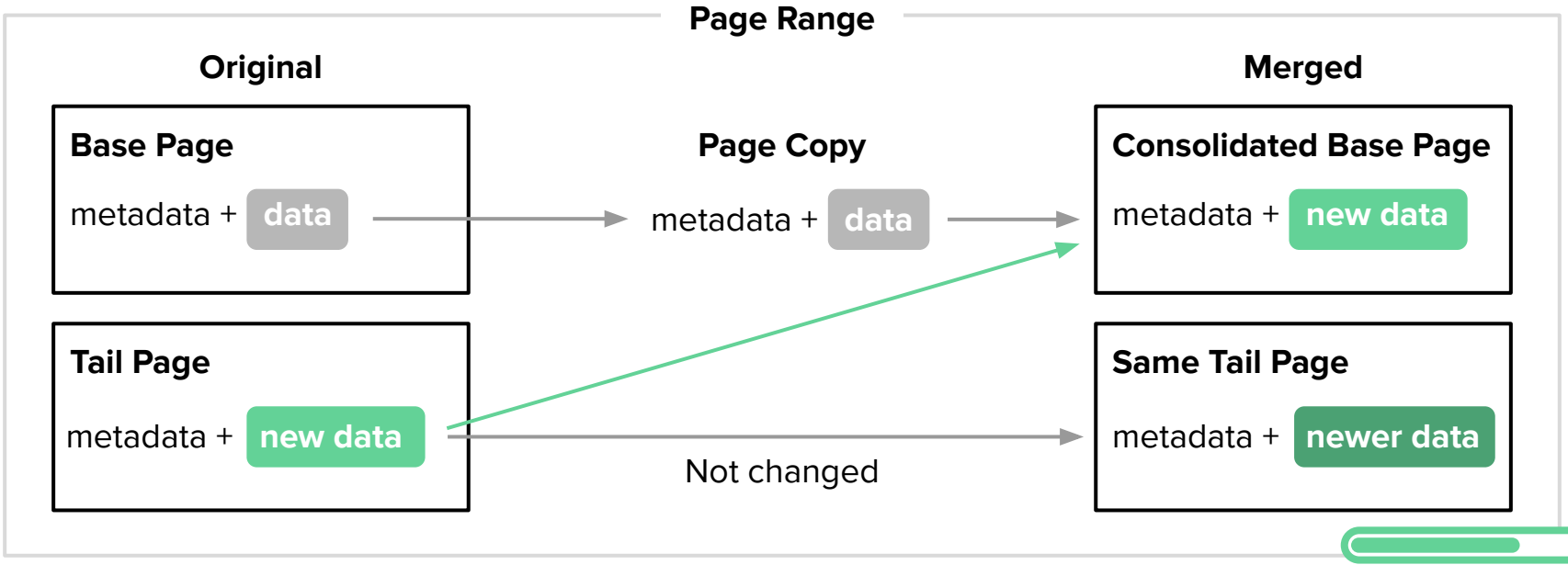
Buffer Pool

- Maintains a list of format [tableName, PR_index_rel_to_table, tailRecordsSinceLastMerge]
- Sorts the table based on the count of tailRecordsSinceLastMerge
- Returns the list with most unmerged tail records to the merge thread

Merge: Method

Consolidates the data of tail pages into the necessary base pages for the entire given PageRange by:

1. Checking every 5 seconds if there are more than 100 changed records
2. Creating a copy of base page(s) but referencing the indirection column to allow for concurrency
3. Changing the data of the new base page instance(s) to the new data in tail record(s)
4. Replacing the original, unmerged base page with the consolidated base page



Metadata: Data Persistence

When closing the DB, there are metadata that needs to **persist** to the next opening of the DB

- In order for the merge thread to know which page range to merge, we keep track of the tailsRecordsSinceLastMerge count even after the DB closes
- In our tables_metadata file we keep track of the metadata for each table, that way if we do new inserts we know the current pageRange, number of columns, and key for each table

page_range_metadata.txt

table_name,page_range_index,tailRecordSinceLastMerge
Grades,0,40000
Grades,1,8000
Grades,2,16000
Grades,3,26444
Grades,4,40000

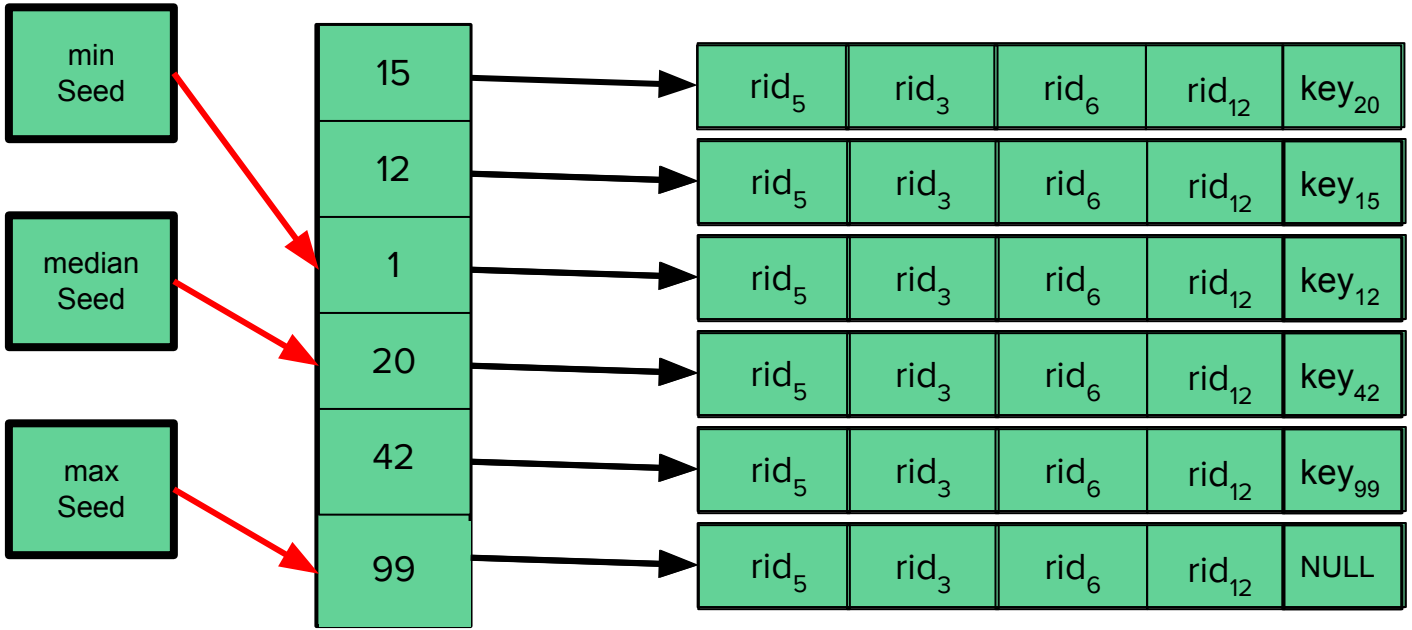
tables_metadata.txt

table_name,num_columns,key,currentPRIndex
Grades,5,0,4

Agenda

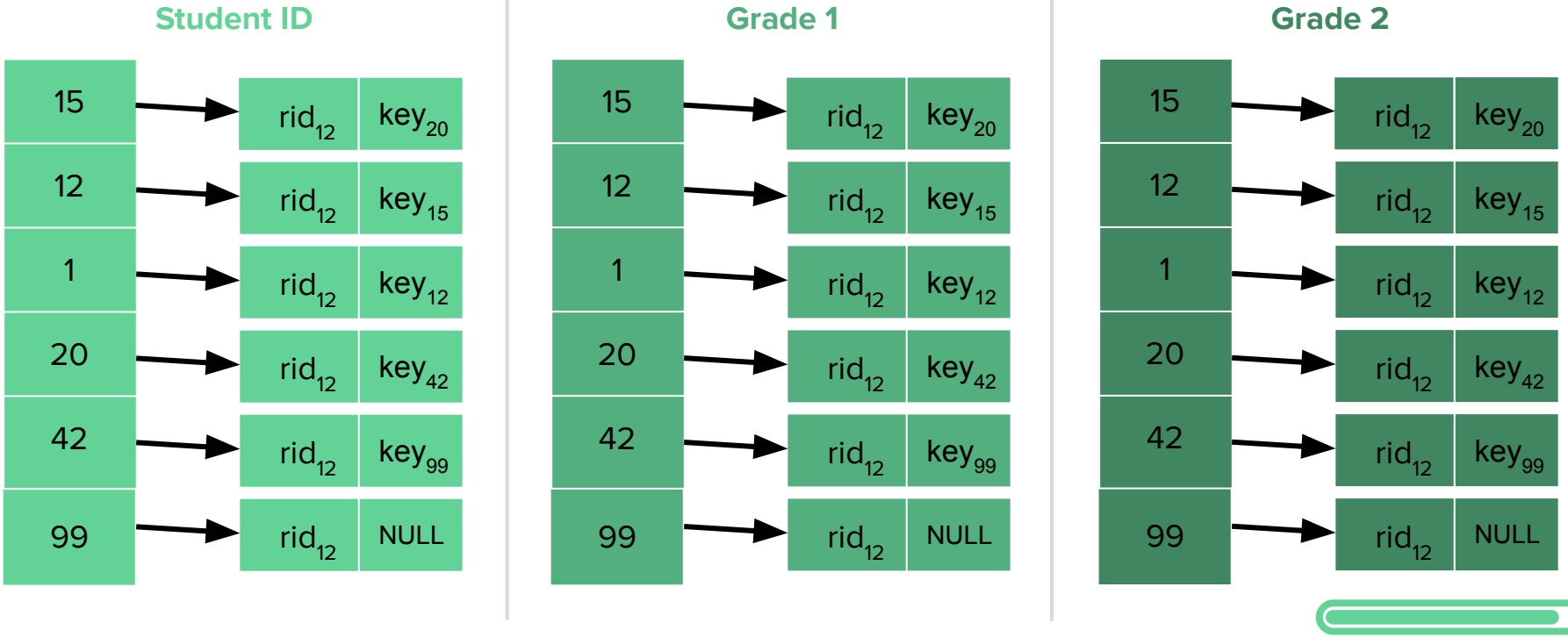
Index: An RHash Implementation

$$\text{index}[\text{key}_k] = [[\text{rids}],\ \text{key}_{k+1}]$$



Index: Multi-Column RHash

RHash indexing can be used on any data column



Tester Runtimes

Number of Records	m2_tester_part1	m2_tester_part2
1,000	8.22 sec	2.85 sec
2,001	15.58 sec	4.06 sec
10,000	1.39.06 min	26.03 sec

Conclusion

Some aspects of the project that we would like to investigate in the next milestone:

- Attempting to merge records by pages instead of page ranges
- Exploring more efficient eviction policies
- Optimizing the buffer pool's runtimes with larger amounts of records

Questions