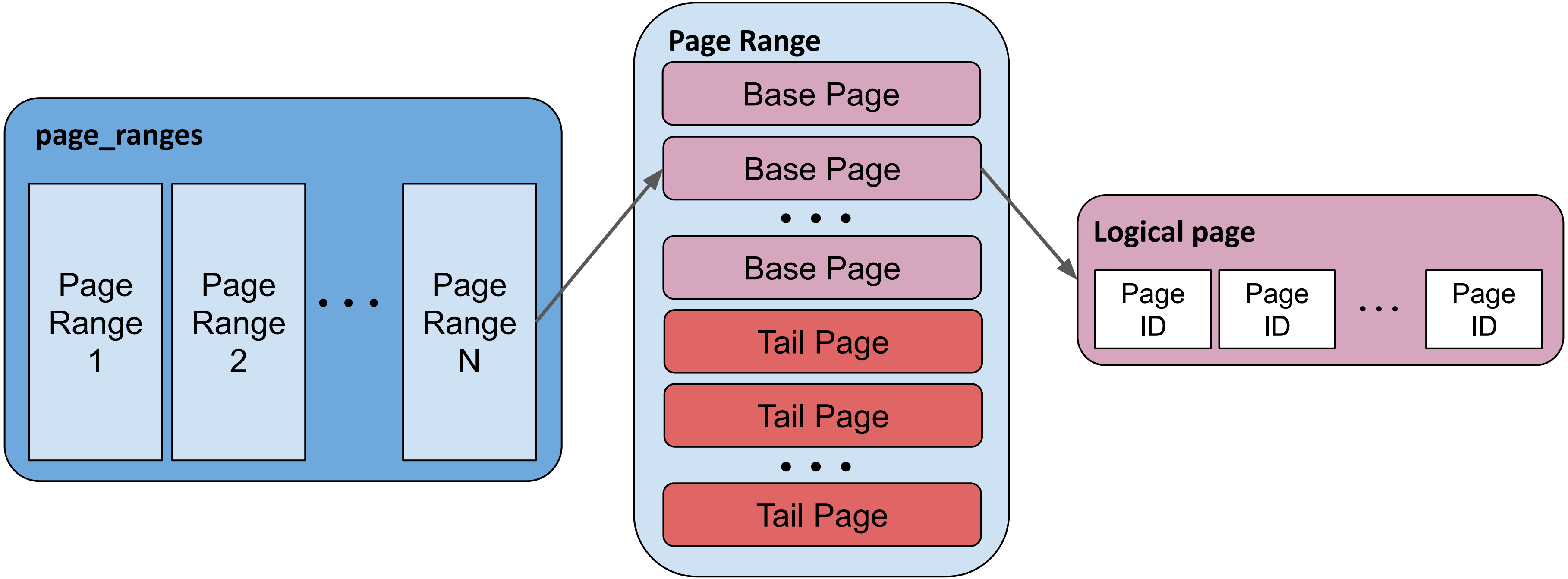
L-Store Milestone 2

Alana Rufer, Eseosa Omorogieva, Nina Gopal,
Riddhi Barbhaiya, Kushaal Rao

Review of logical memory organization

RID ⟶ [page_range_offset, page_index, offset]



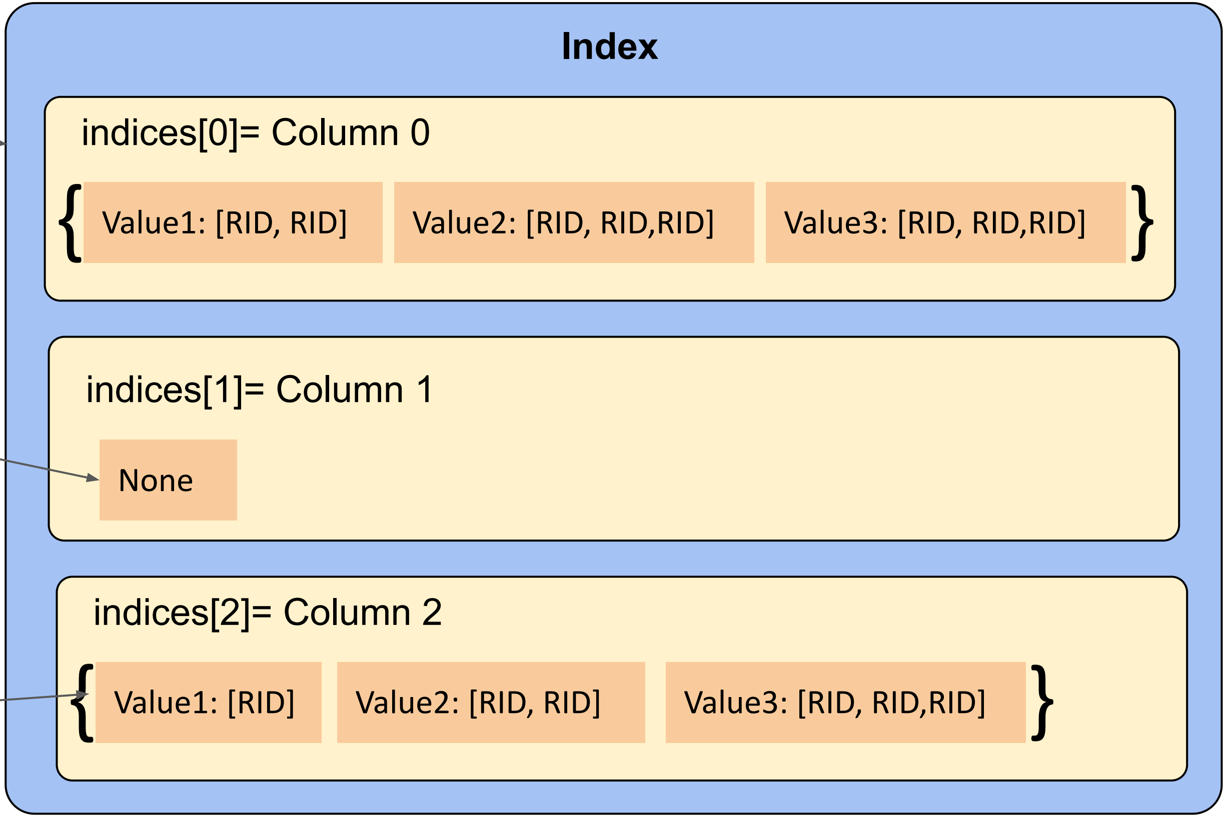
Indexing

Index Structure

Array of dictionaries, with the index of each dictionary corresponding to the column number

Columns without indexes have a None placeholder

Value are mapped to a list of RIDs of records that contain the value in that column



Index

indices[0]= Column 0

{ Value1: [RID, RID] Value2: [RID, RID,RID] Value3: [RID, RID,RID] }

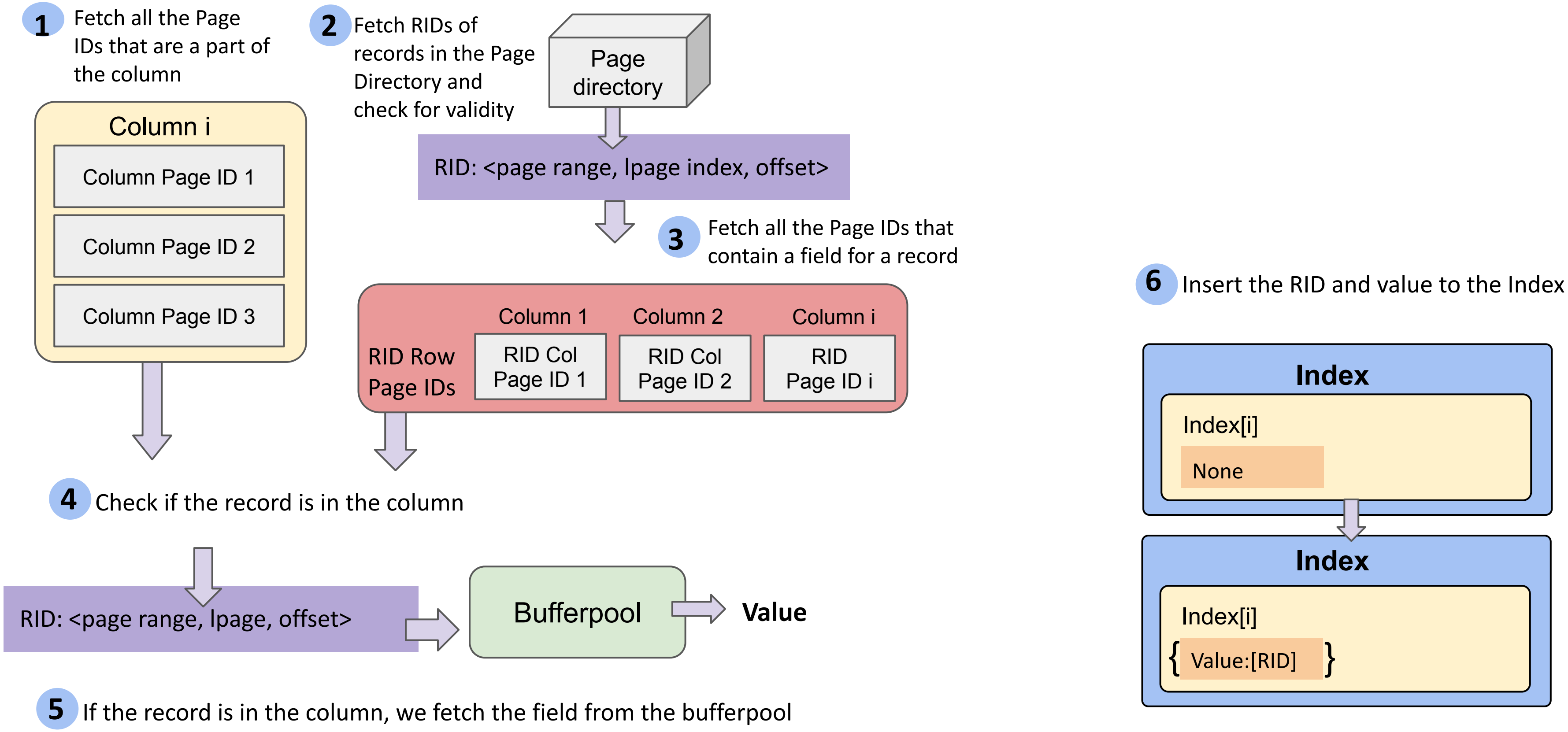
indices[1]= Column 1

None

indices[2]= Column 2

{ Value1: [RID] Value2: [RID, RID] Value3: [RID, RID,RID] }

Index Create and Drop Column



1 Fetch all the Page IDs that are a part of the column

Column i

Column Page ID 1

Column Page ID 2

Column Page ID 3

2 Fetch RIDs of records in the Page Directory and check for validity

Page directory

RID: <page range, lpage index, offset>

3 Fetch all the Page IDs that contain a field for a record

	Column 1	Column 2	Column i
RID Row Page IDs	RID Col Page ID 1	RID Col Page ID 2	RID Page ID i

4 Check if the record is in the column

RID: <page range, lpage, offset>

Bufferpool

Value

5 If the record is in the column, we fetch the field from the bufferpool

6 Insert the RID and value to the Index

Index

Index[i]

None

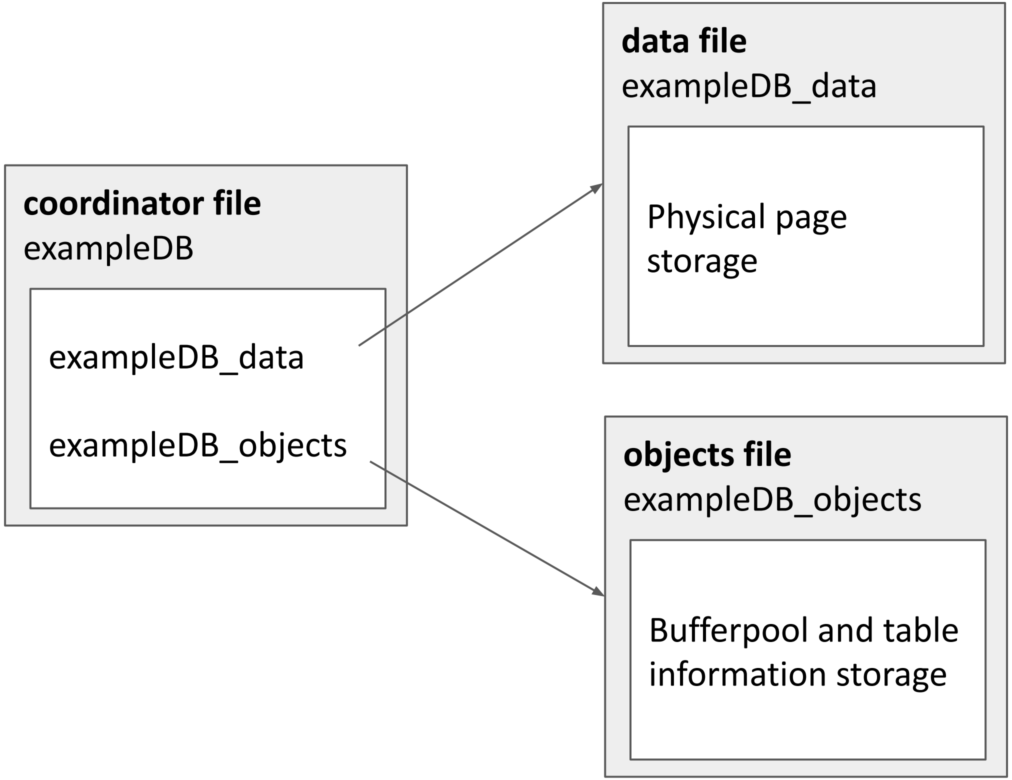
Index

Index[i]

{ Value:[RID] }

Durability

File Organization & Structure



coordinator file
exampleDB

exampleDB_data

exampleDB_objects

data file
exampleDB_data

Physical page storage

objects file
exampleDB_objects

Bufferpool and table information storage

3 files

Separate files for data and for table & bufferpool information

db.open(): restore from files

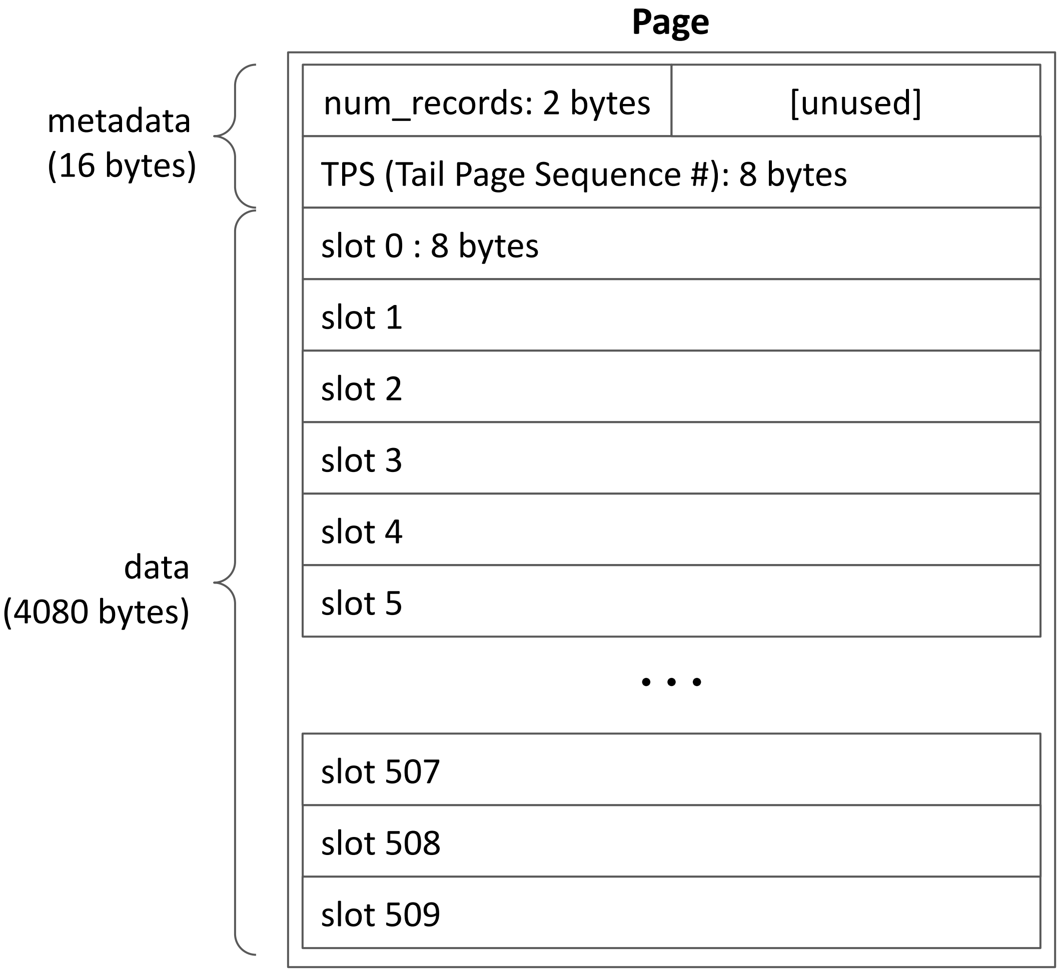
db.close(): flush dirty pages to data file and store other information in objects file

Physical Page Metadata

Moved metadata inside the 4096 block allocated to a physical page

Added **TPS** (Tail Page Sequence Number): RID of last merged tail record

TPS is used to help determine whether a field is up to date



Page

metadata (16 bytes)

data (4080 bytes)

num_records: 2 bytes	[unused]
TPS (Tail Page Sequence #): 8 bytes	
slot 0 : 8 bytes	
slot 1	
slot 2	
slot 3	
slot 4	
slot 5	
. . .	
slot 507	
slot 508	
slot 509	

Bufferpool

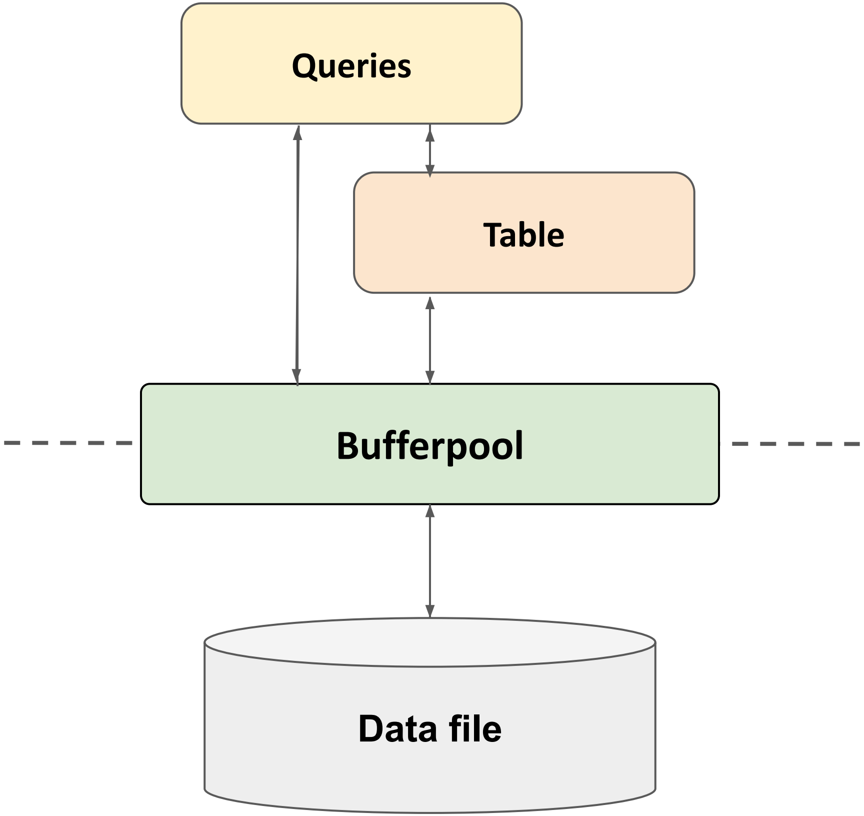
Abstracts disk accesses

Shared by all database tables

Granularity of physical pages

Uses locks to prevent conflict between merge and user threads

Deallocates disk space when a table is dropped

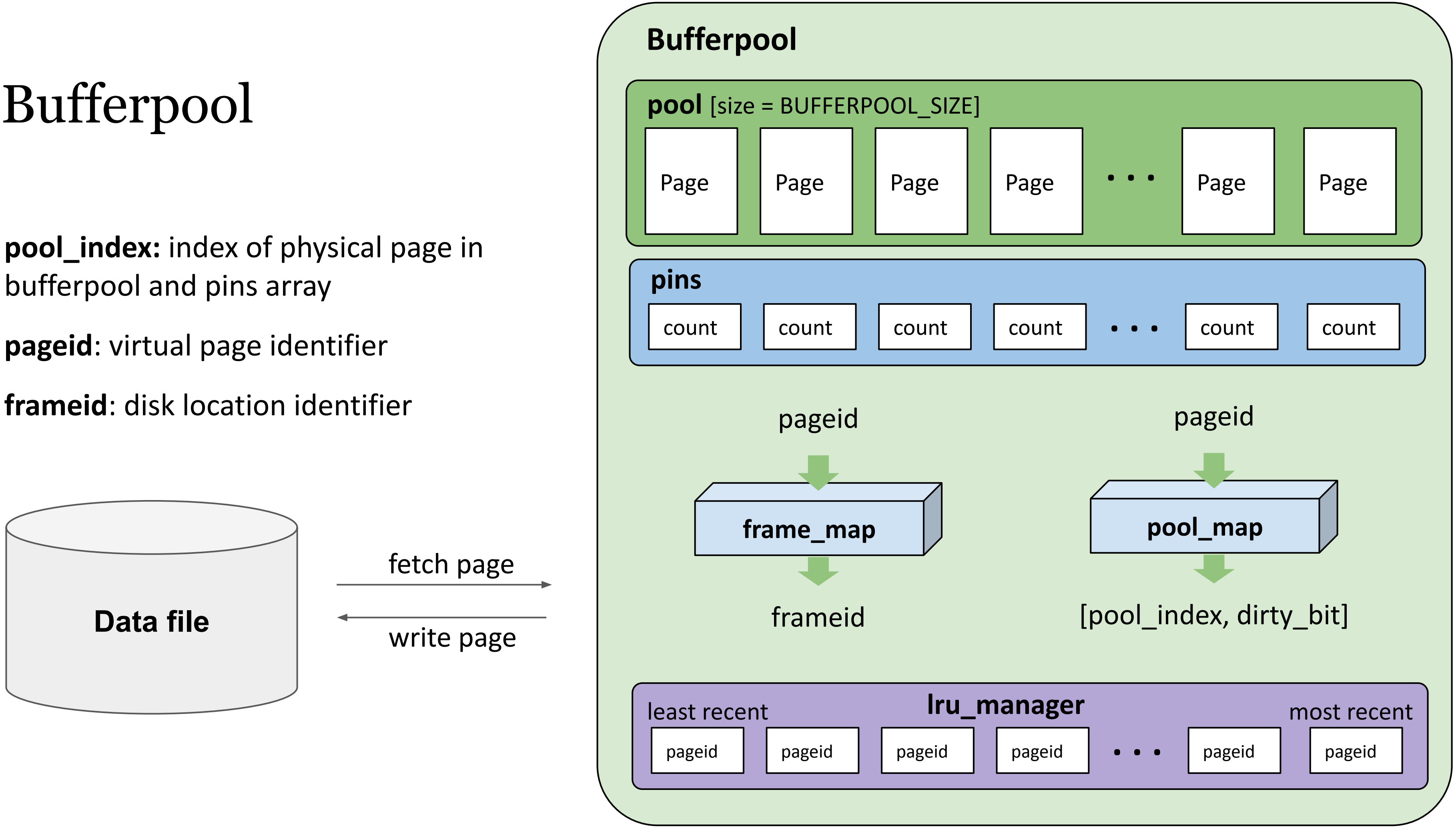


Bufferpool

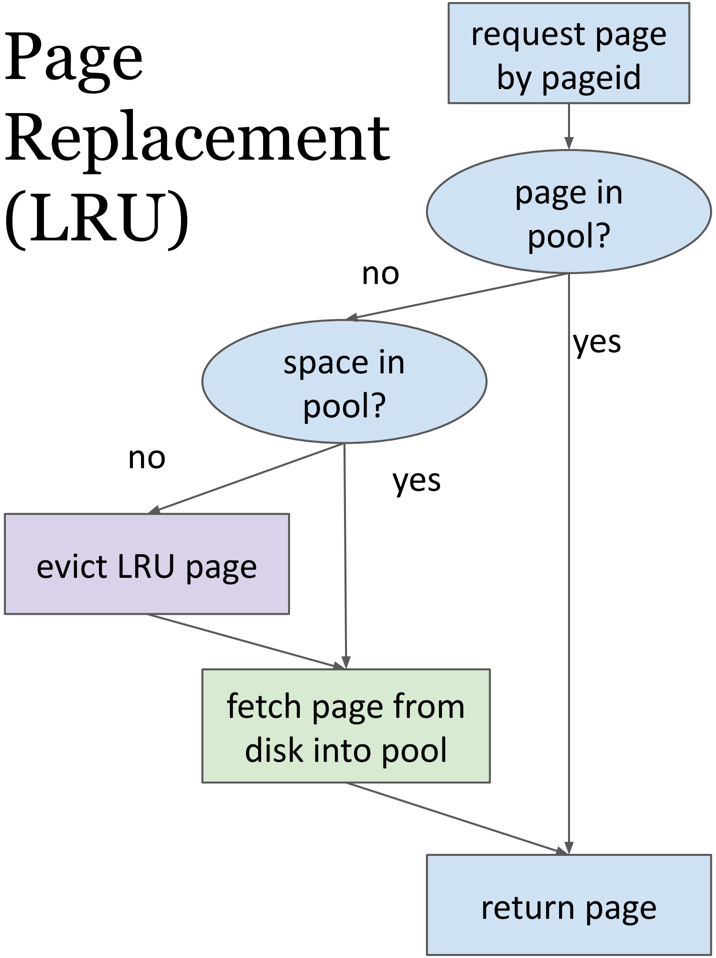
pool_index: index of physical page in bufferpool and pins array

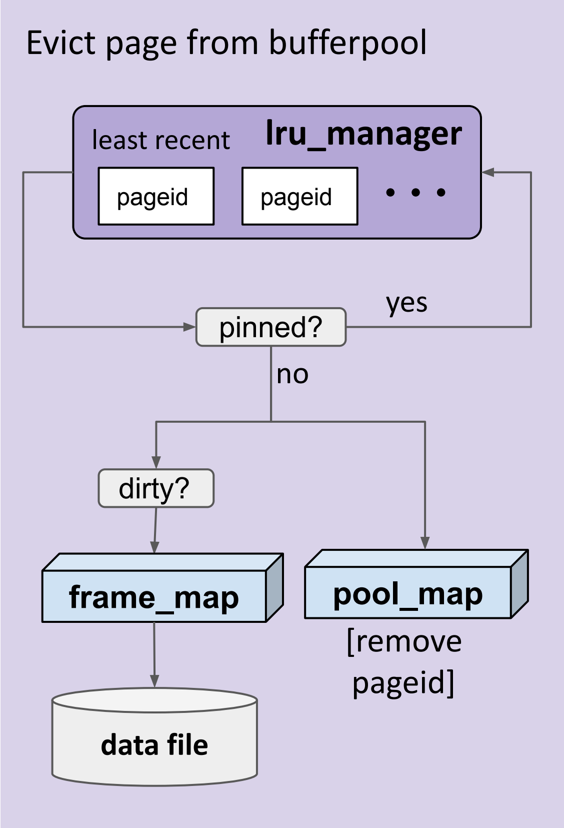
pageid: virtual page identifier

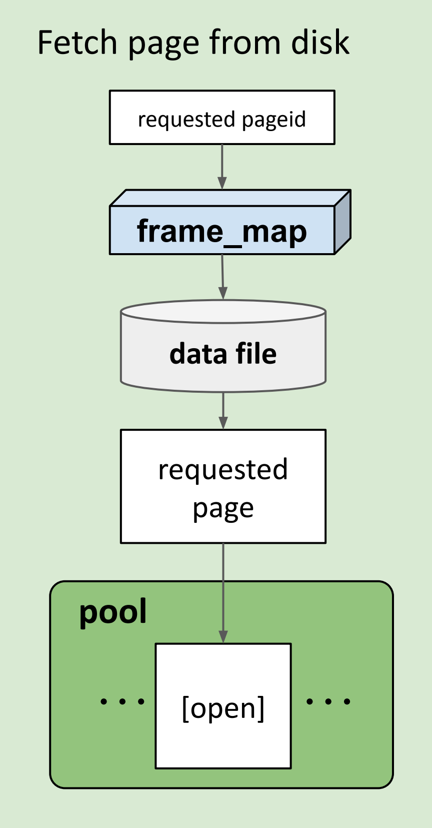
frameid: disk location identifier



Page Replacement (LRU)

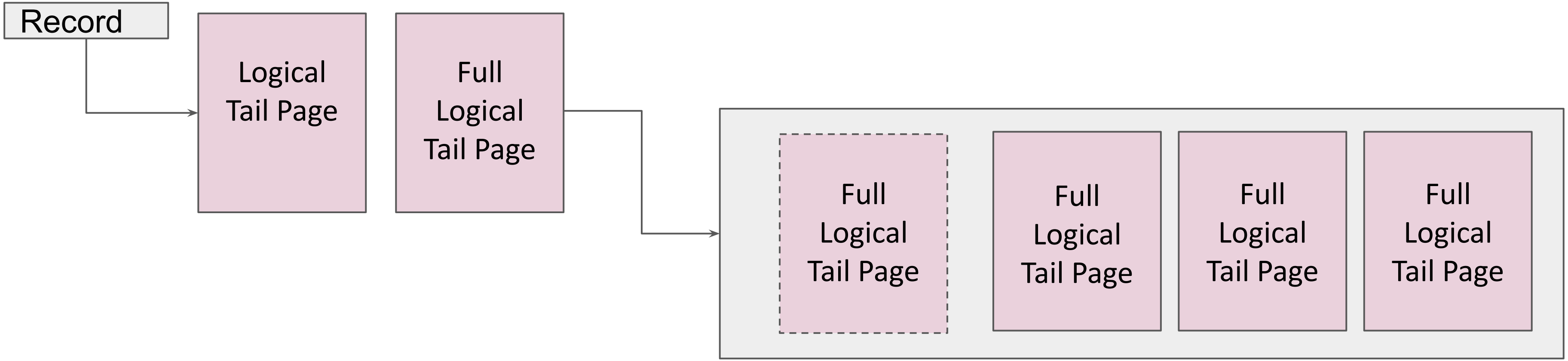



Evict page from bufferpool


Fetch page from disk

Merging

Initiating Merge

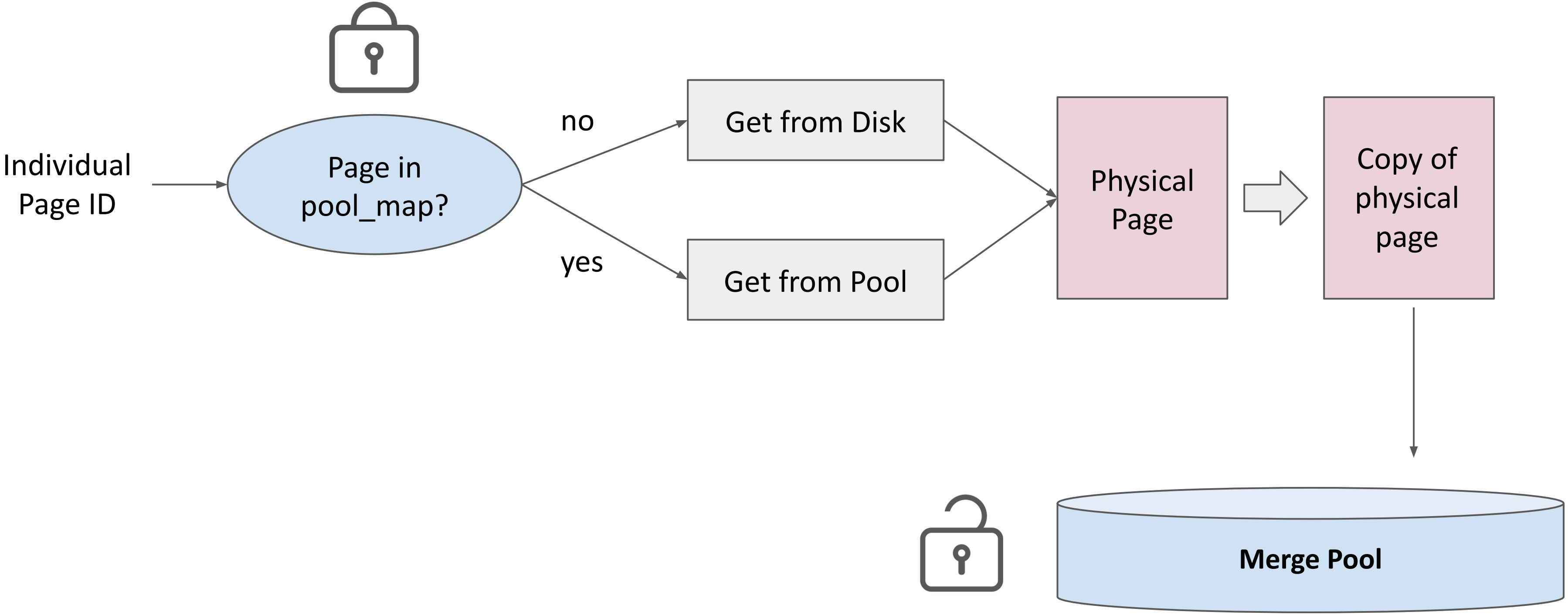


Merge Queue (copies only)

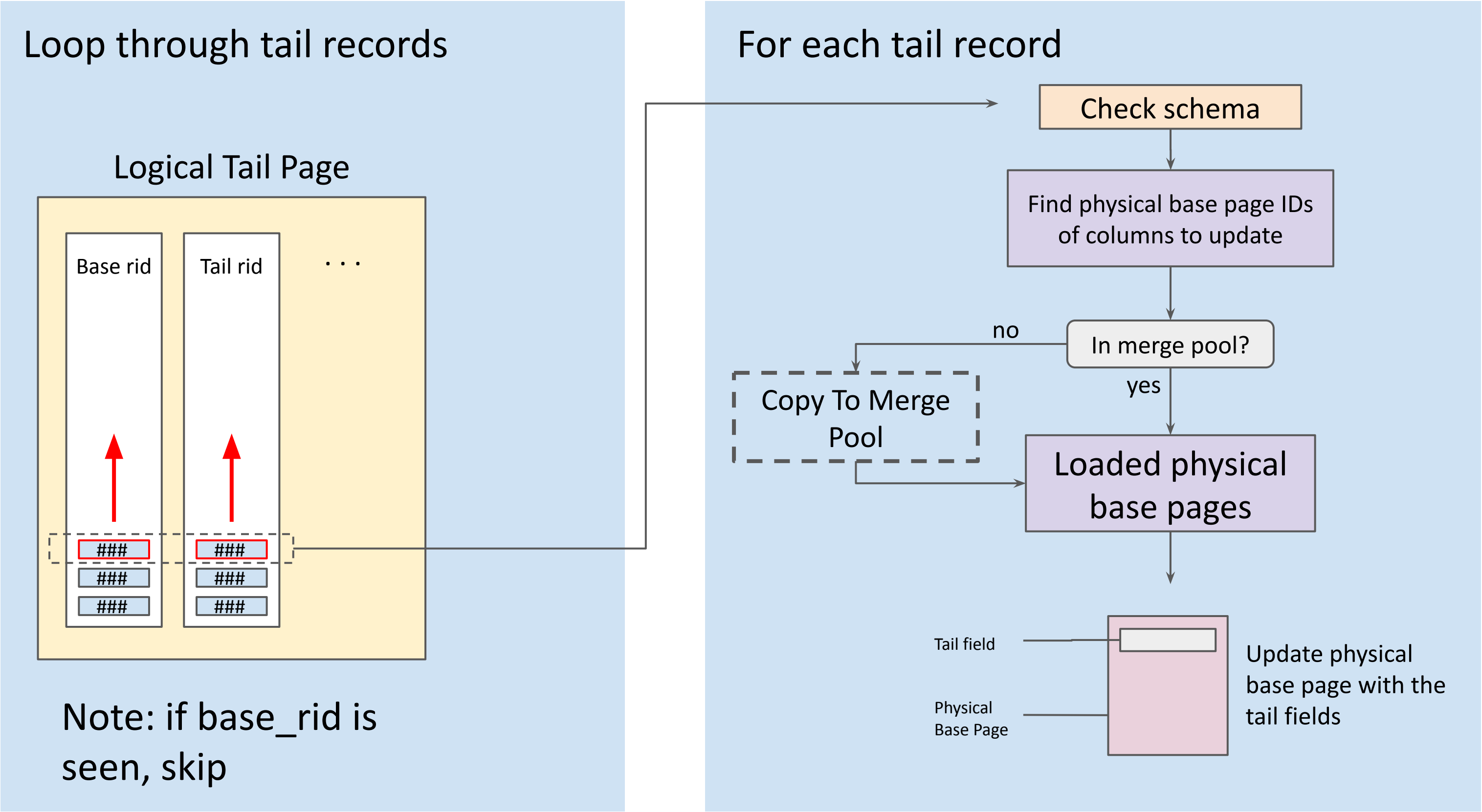
Merge after reaching threshold value of filled tail pages

Copy To Merge Pool

Separate storage area for pages used in the merge



Merging



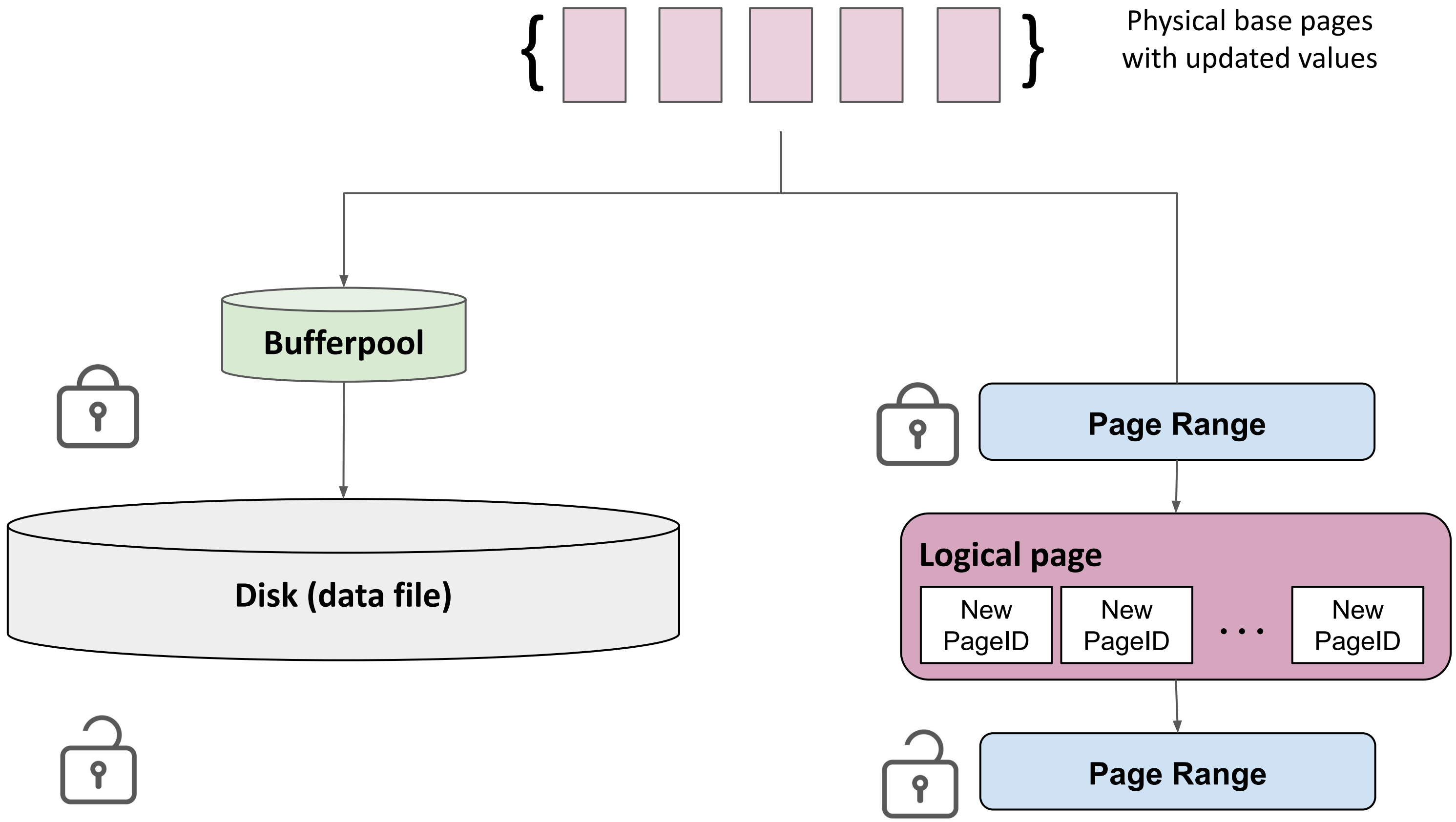
Loop through tail records

Note: if base_rid is seen, skip

For each tail record

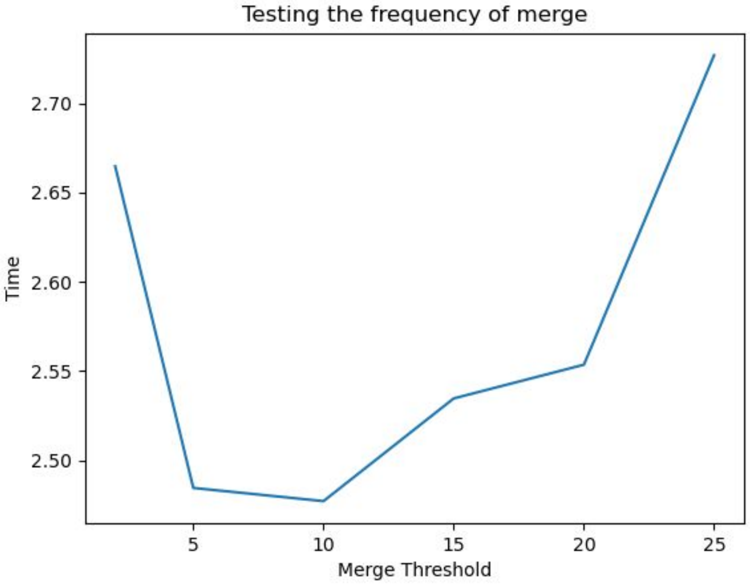
Update physical base page with the tail fields

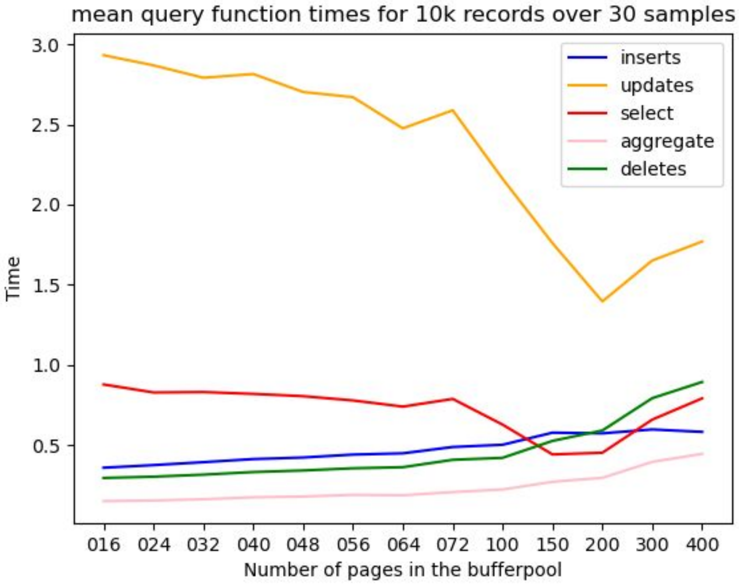
Concluding Merge



Performance

Tuning Bufferpool and the Threshold for Merging



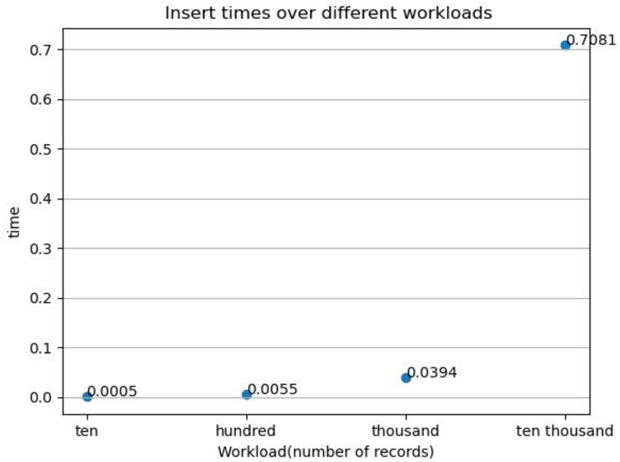


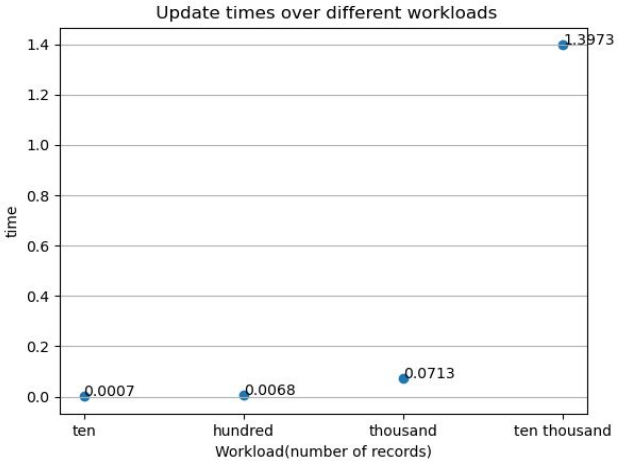
Finalized Merge Threshold = 10

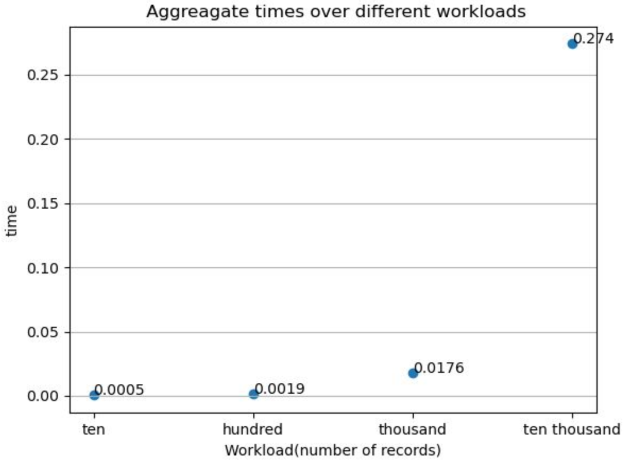
Finalized Bufferpool = 200

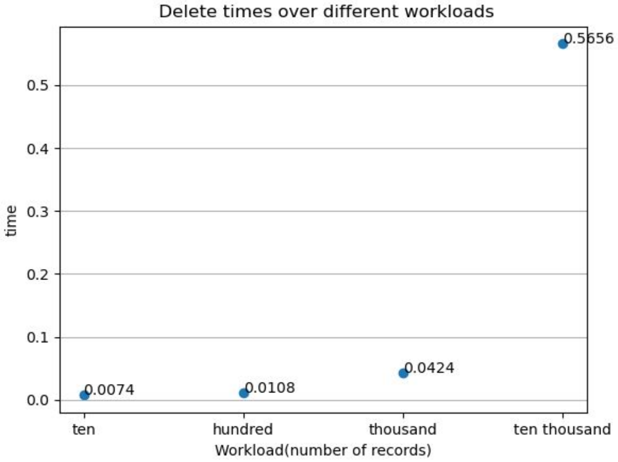
Workload: tuning_merge.py, tuning_buffer.py
Hardware: Dual-Core Intel Core i7, 2.5GHz, 16GB, 4 MB L3 Cache

Query performance









Workload: __main__.py

Hardware: Dual-Core Intel Core i7, 2.5GHz, 16GB, 4 MB L3 Cache

Q&A