16 February 2024

Milestone 1: Design, Accomplishments, and Solutions

Ashwin Chembu, Ruohan Huang, Trevor Lash, Mira Stenger, Yuecheng Zhao

TEAM

Yuecheng Zhao

Mira Stenger

Ruohan Huang

Trevor Lash

Ashwin Chembu

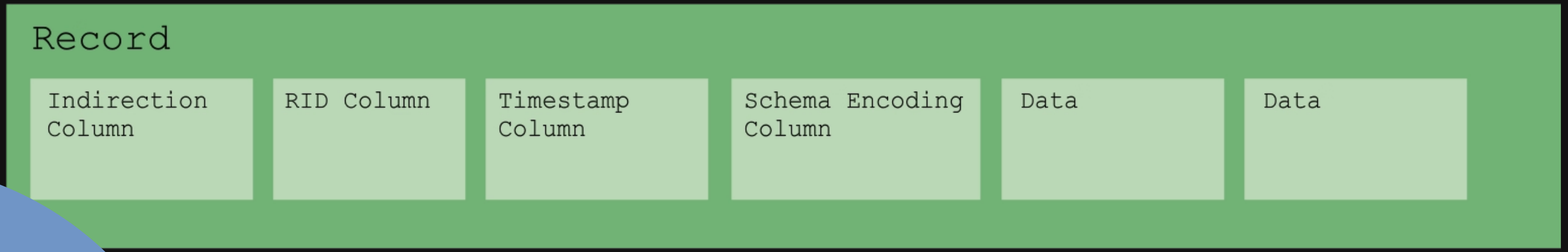
Data Model

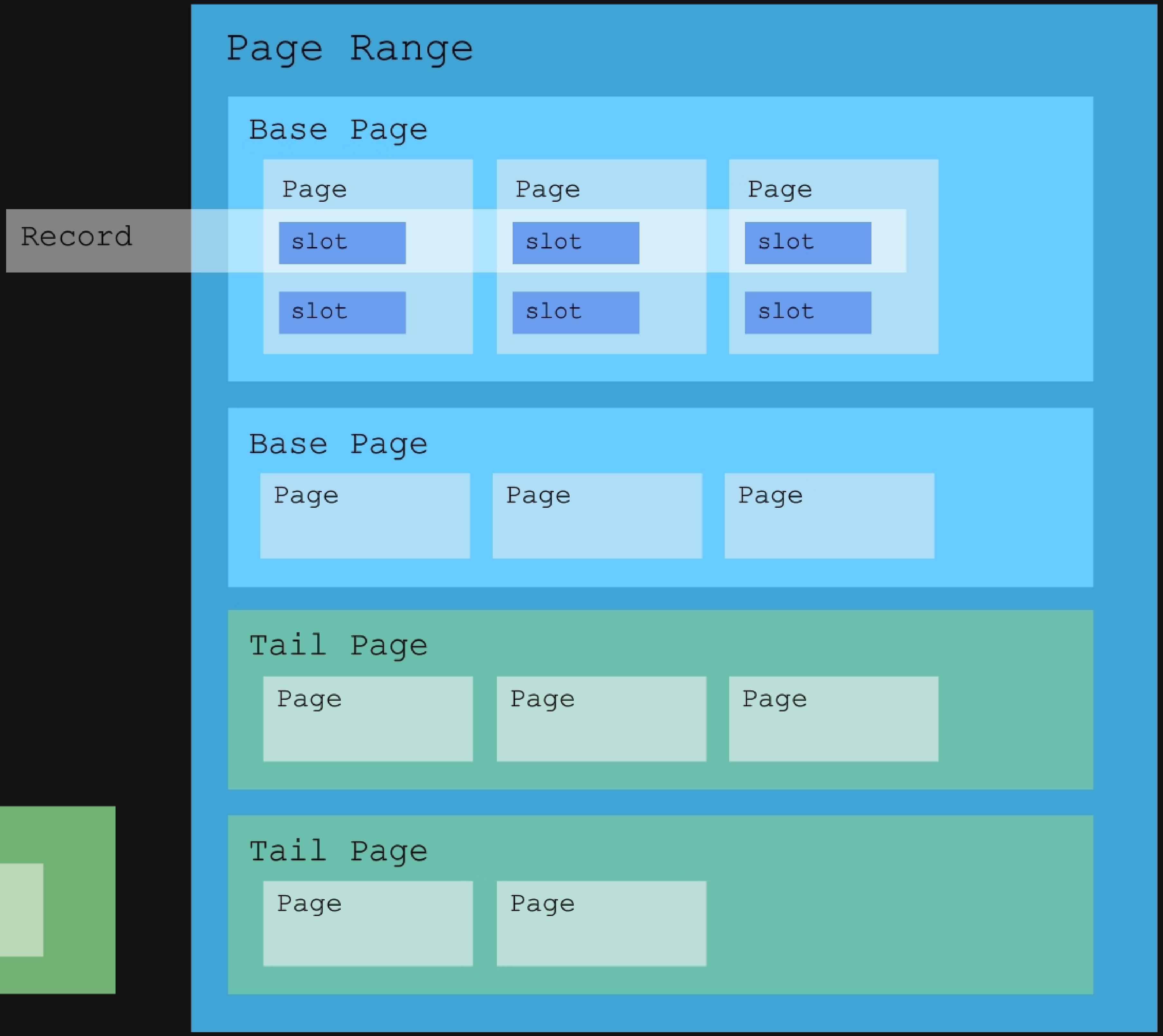
Page Range

- main purpose of abstraction
- Consists of base pages and corresponding tail pages
- Updates for all records are appended to the same tail page

Page

- 4096 byte page size
- fixed length records and packed page format





Bufferpool Management

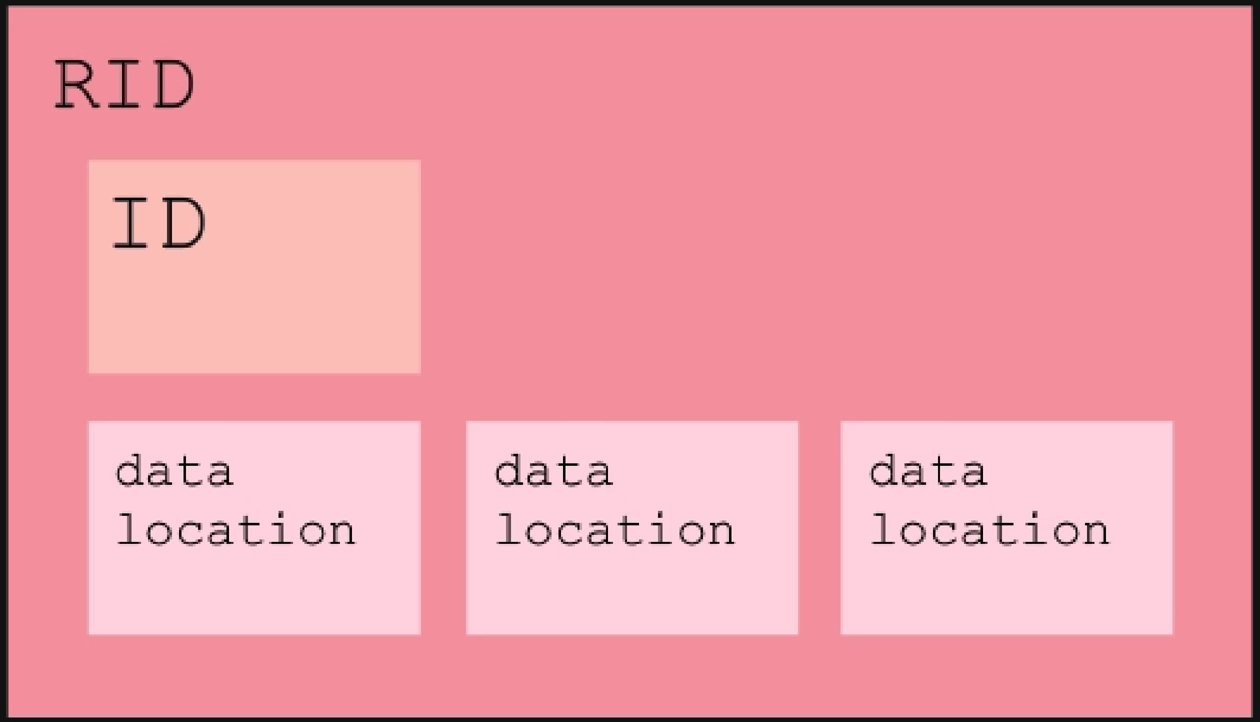
16 February 2024

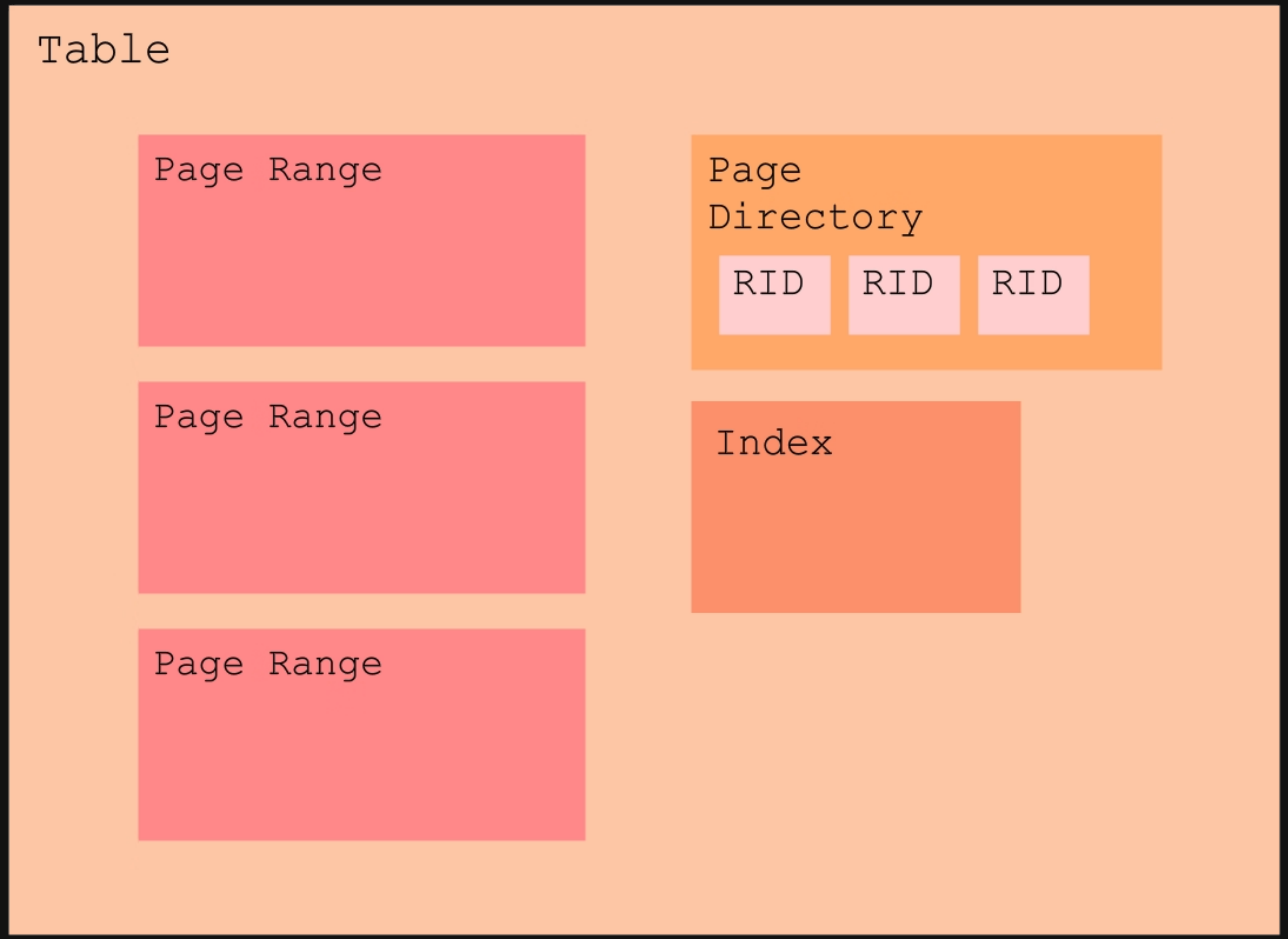
RID

- contains a numerical ID
- vector of pointers to metadata and data

Table

- vector of page ranges to store data
- page directory to map RID IDs to instances of the RID class
- pointer to index





Query Interface

Insert

- calls the table's & table's index's insert function to write

Select

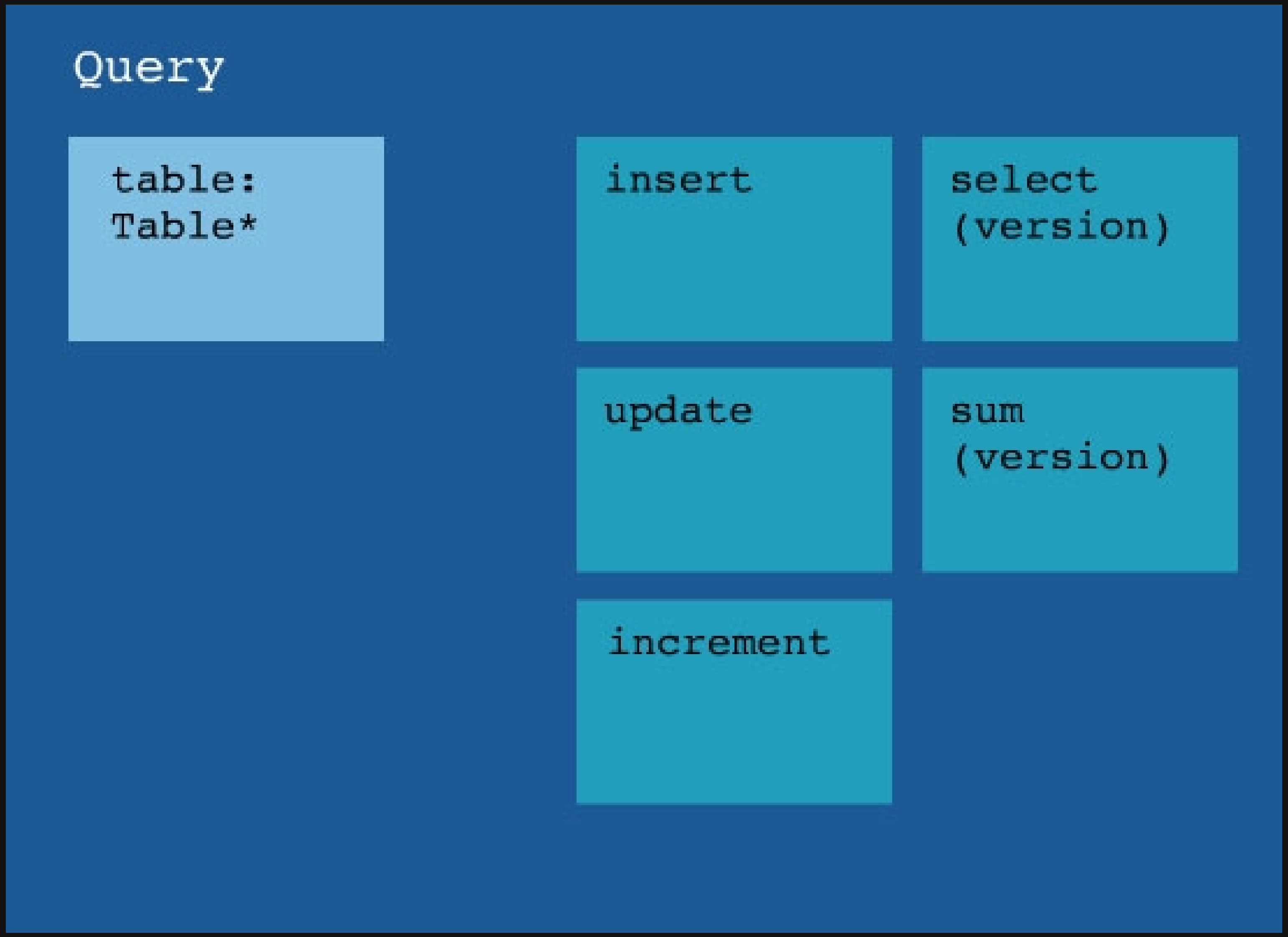
- gets list of RID from index with most recent version(or given version) into record object, returns array of record objects

Update

- locates RID to be updated
- adds to tail page(or redirects)
- updates indirection column and index

Sum

- locates required range of RID using the index
- navigates the indirection column(depending on versioning)
- returns sum



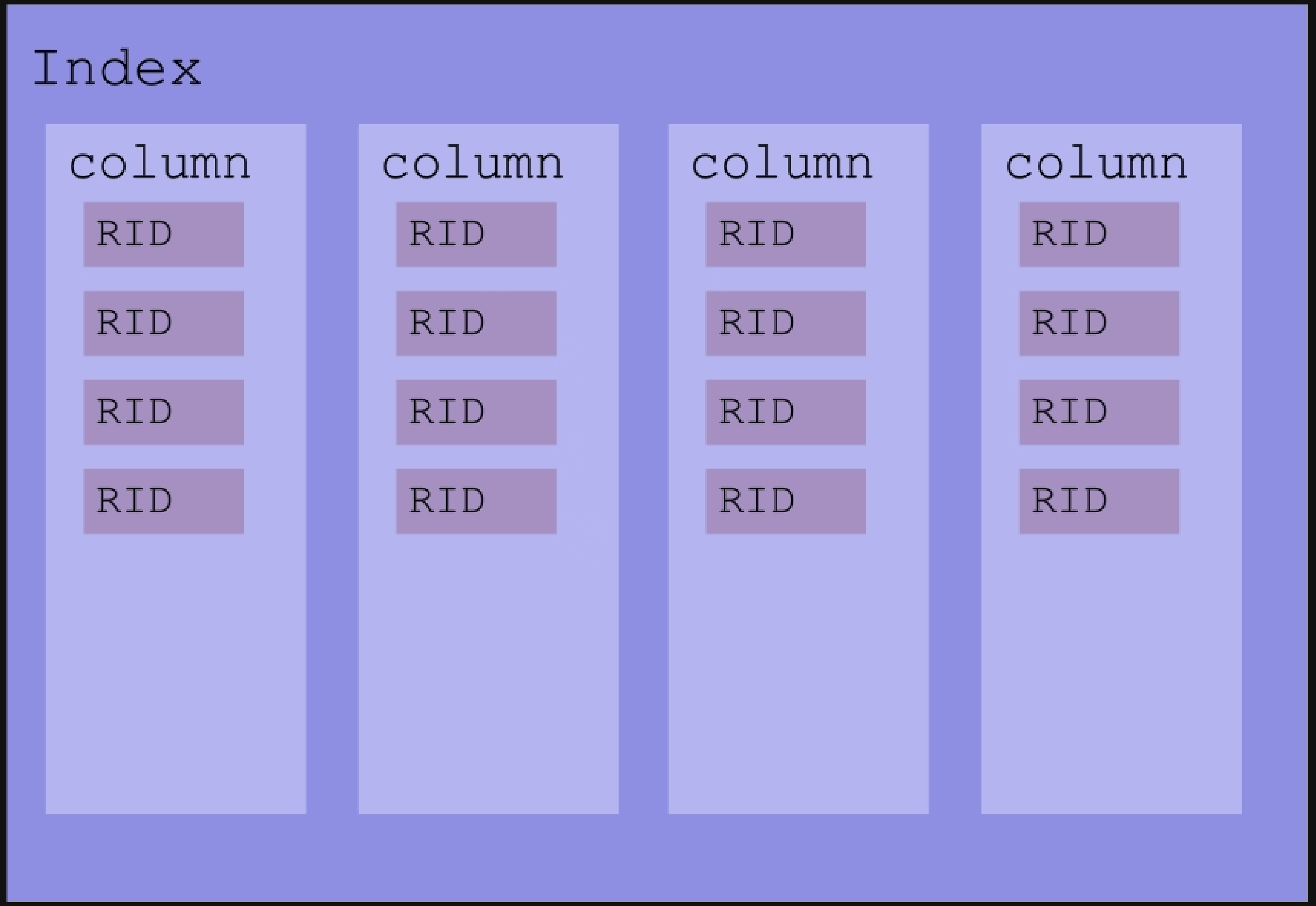
Increment

- gets the locates the key
- iterates through the associated columns
- increments it

Index

indexing

- unordered multi-maps for efficient index storage and quick record location
- key → data in the columns
- value → RID
- allows for indexing on each column(not just key)



Python Wrapper

Compilation:

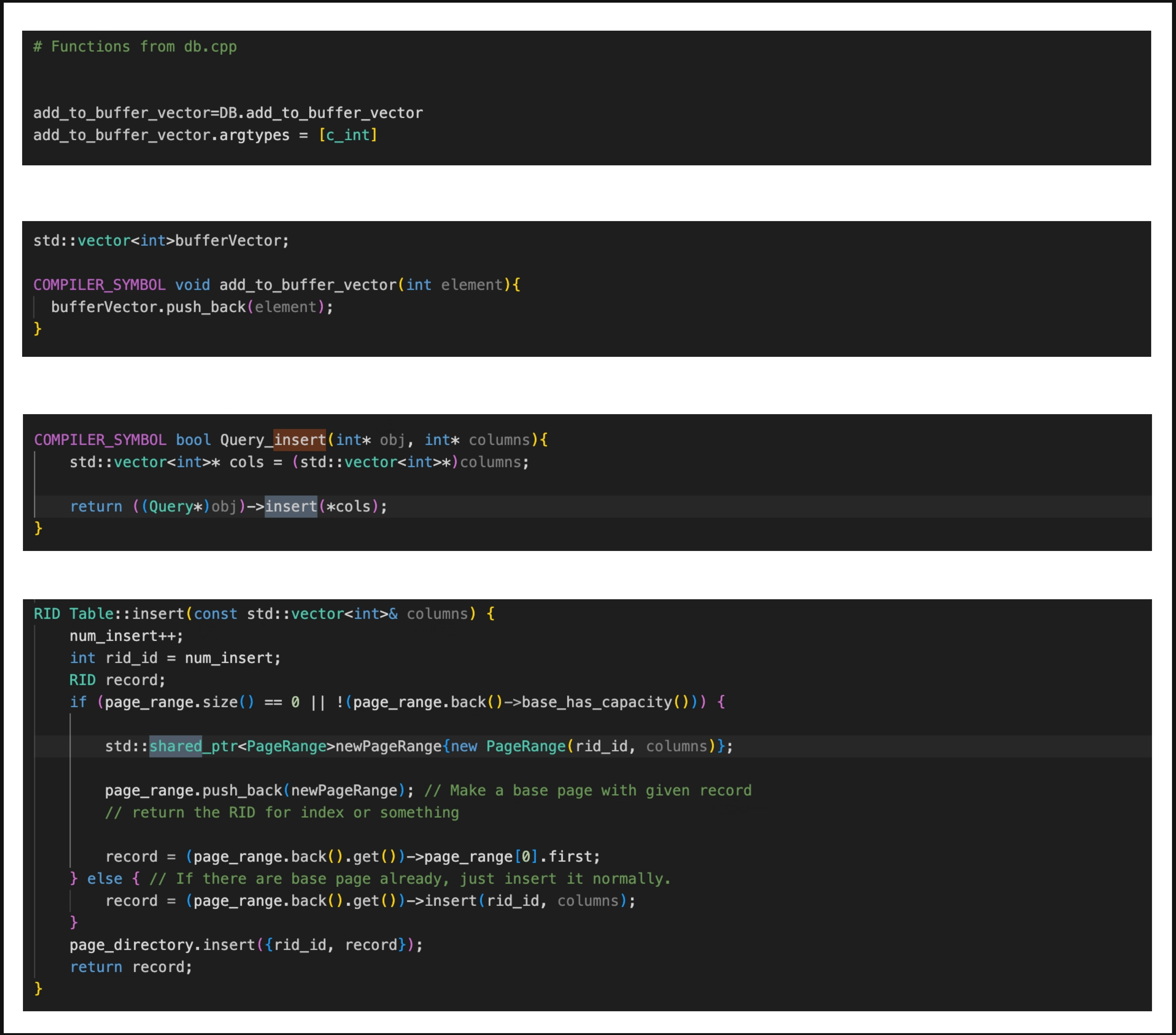
- C++ functions are compiled into .so, .dylib, and .dll files using a makefile for ctypes access.
- Ctypes restricts C++ functions to pointers and primitives.

Ctypes Workarounds:

- Global arrays are used to parse Python lists in query.py and query.cpp.
- The Query.insert method demonstrates data passing between Python and C++.

Lifetime Management:

- Table class uses shared_ptrs for PageRanges to prevent memory leaks.
- Ensures resources are not released until all references are gone.



```
# Functions from db.cpp


add_to_buffer_vector=DB.add_to_buffer_vector
add_to_buffer_vector.argtypes = [c_int]
```

```
std::vector<int>bufferVector;

COMPILER_SYMBOL void add_to_buffer_vector(int element){
  bufferVector.push_back(element);
}
```

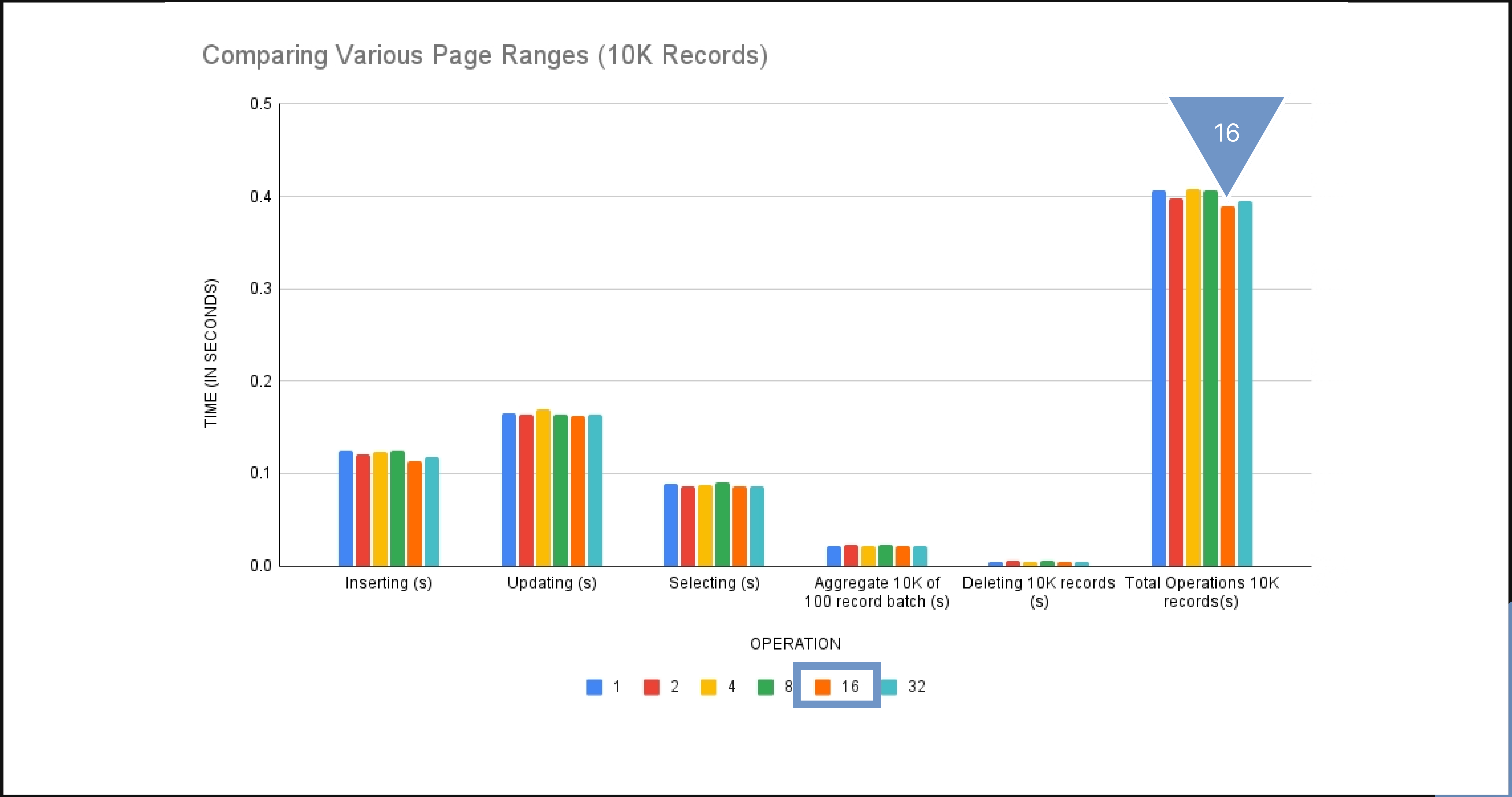
```
COMPILER_SYMBOL bool Query_insert(int* obj, int* columns){
    std::vector<int>* cols = (std::vector<int>*)columns;

    return ((Query*)obj)->insert(*cols);
}
```

```
RID Table::insert(const std::vector<int>& columns) {
    num_insert++;
    int rid_id = num_insert;
    RID record;
    if (page_range.size() == 0 || !(page_range.back()->base_has_capacity())) {

        std::shared_ptr<PageRange>newPageRange{new PageRange(rid_id, columns)};

        page_range.push_back(newPageRange); // Make a base page with given record
        // return the RID for index or something

        record = (page_range.back().get())->page_range[0].first;
    } else { // If there are base page already, just insert it normally.
        record = (page_range.back().get())->insert(rid_id, columns);
    }
    page_directory.insert({rid_id, record});
    return record;
}
```

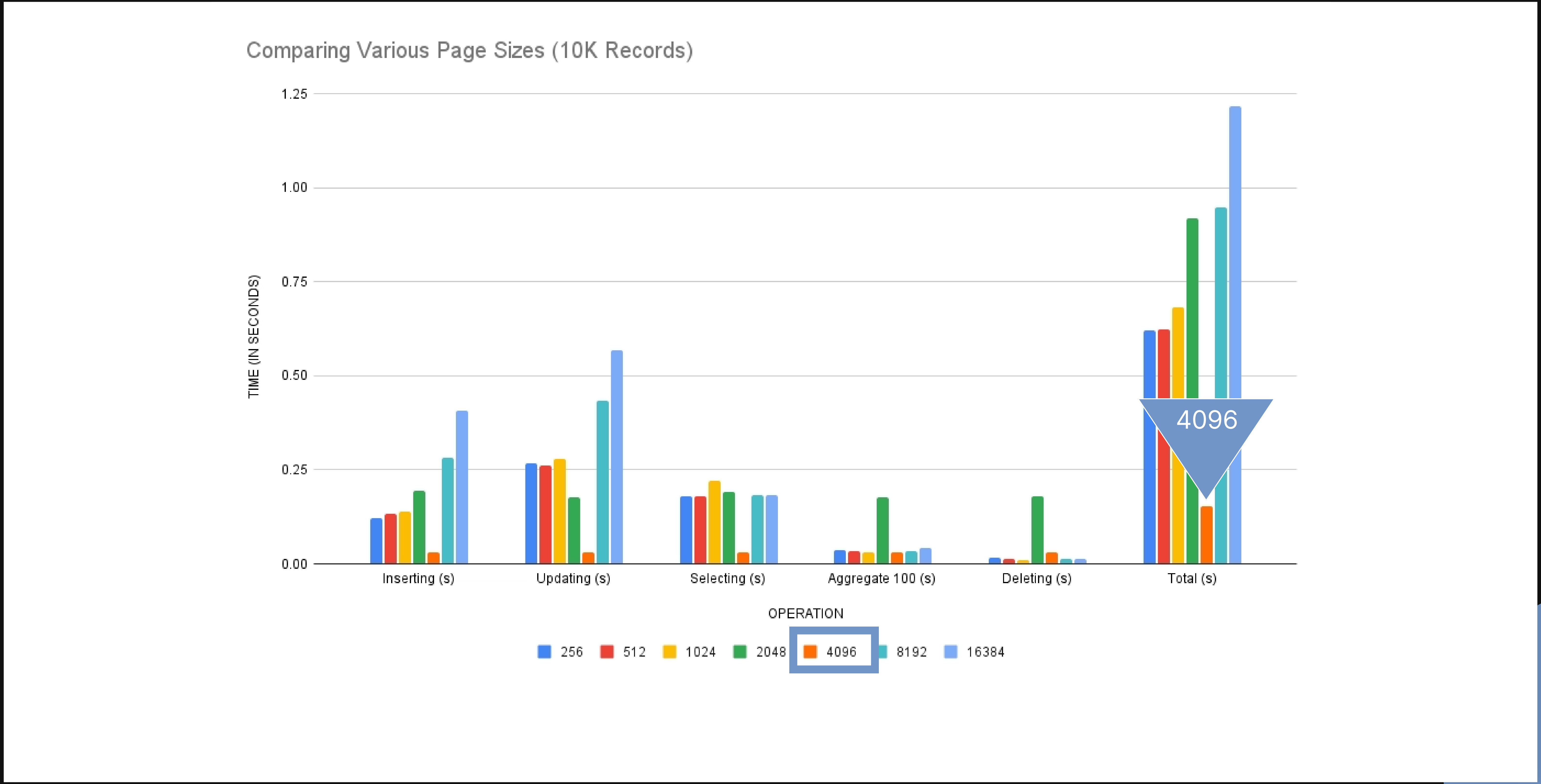
Performance

Inserting 10K records	0.12386
Updating 10K records	0.170233
Selecting 10K records	0.088323
Aggregate 10K of 100 record batch	0.021175
Deleting 10K records	0.005082

Graphs


Comparing Various Page Ranges (10K Records)
TIME (IN SECONDS)
0.5
0.4
0.3
0.2
0.1
0.0
16
Inserting (s)
Updating (s)
Selecting (s)
Aggregate 10K of 100 record batch (s)
Deleting 10K records (s)
Total Operations 10K records(s)
OPERATION
1
2
4
8
16
32

Graphs


Comparing Various Page Sizes (10K Records)
TIME (IN SECONDS)
1.25
1.00
0.75
0.50
0.25
0.00
Inserting (s)
Updating (s)
Selecting (s)
Aggregate 100 (s)
Deleting (s)
Total (s)
OPERATION
256
512
1024
2048
4096
8192
16384
4096

Thank you