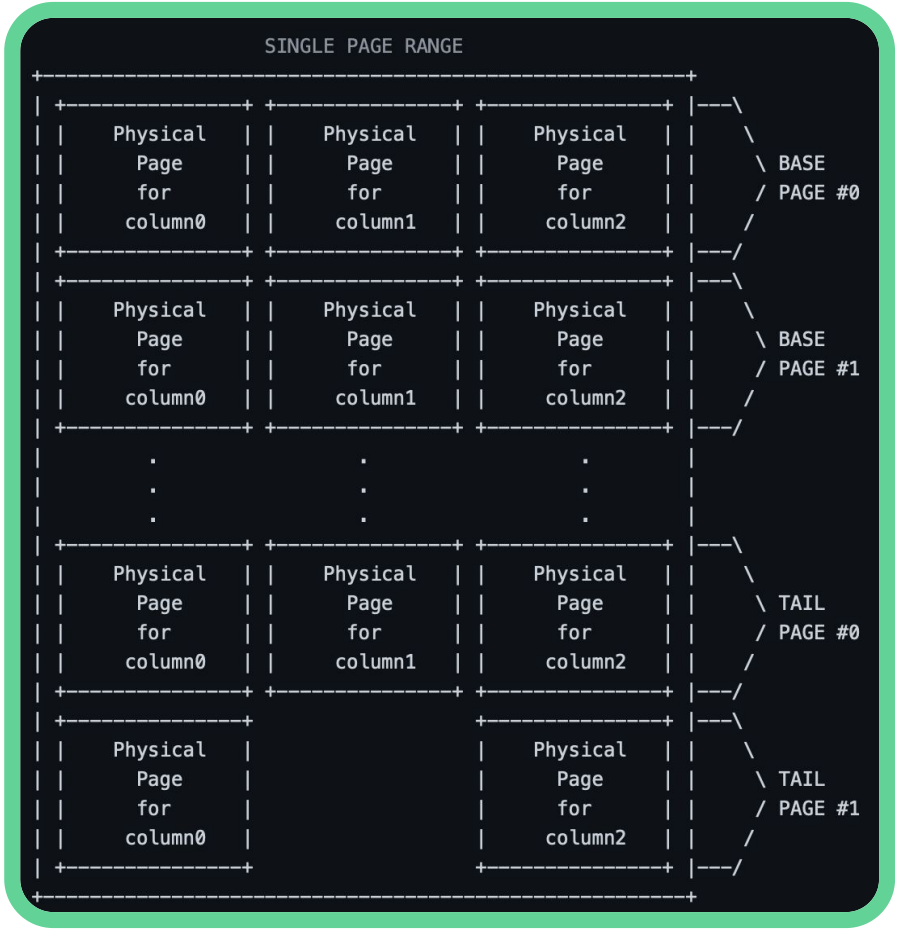
# Milestone 1: Single-threaded, In-memory L-Store

ECS165A - WQ2021

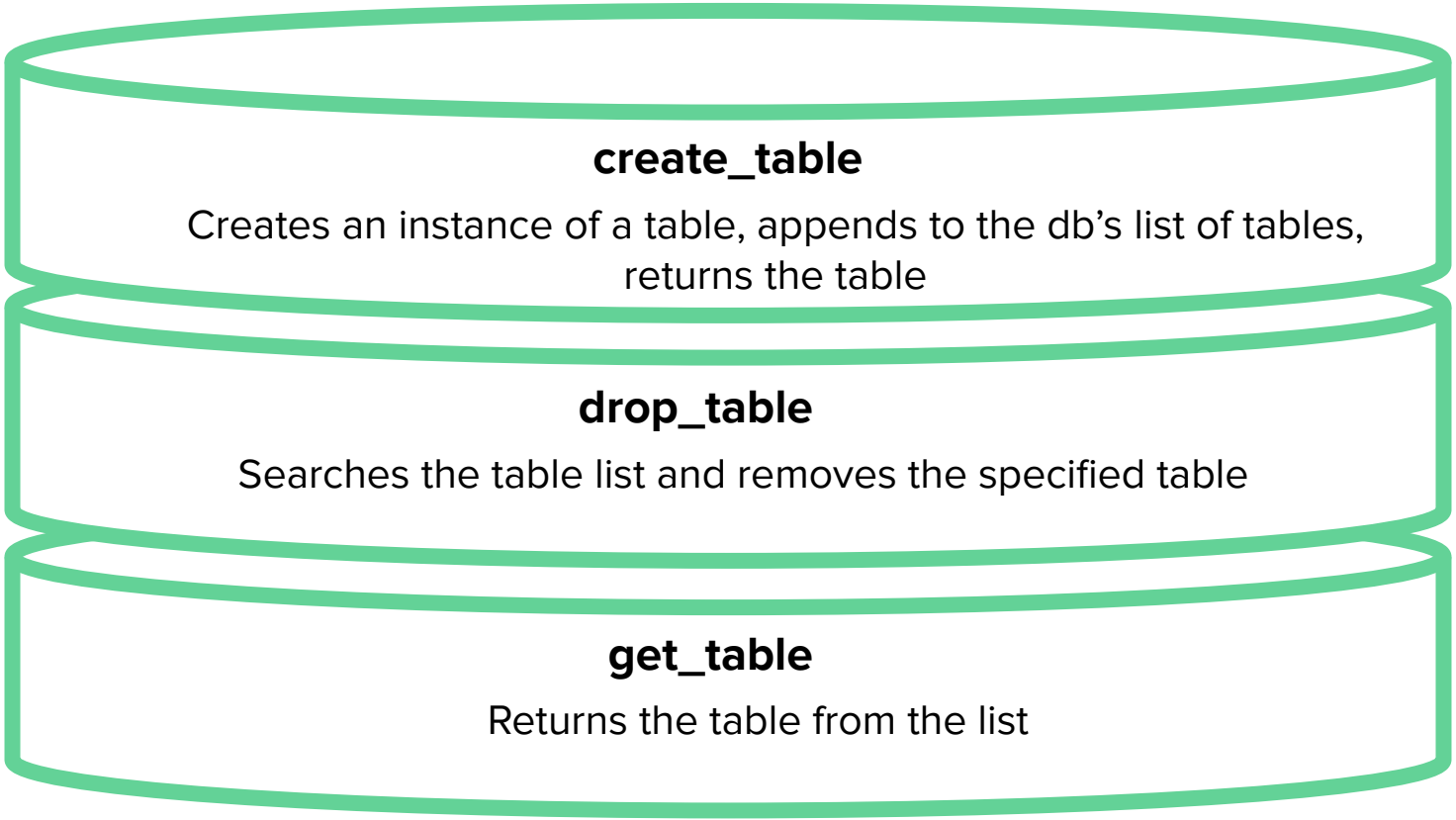
Sean Carnahan, Reese Lam, Gabriel Vazquez,
Quang-Long Tran, Aly Kapasi

# Introduction

- Designed a Single-Thread In-Memory L-Store Database
- Columnar Database containing Pages as columns
- Uses Base and Tail Pages to store and update data
- Page Ranges contain record ranges of the column
- Base Pages are read-optimized while the Tail Pages are write-optimized
- RHash Implementation for Indexing



# Database


create_table
Creates an instance of a table, appends to the db's list of tables, returns the table
drop_table
Searches the table list and removes the specified table
get_table
Returns the table from the list

# Table: Architecture

**Page Directory**

Page Range 1

Page Range 2

Uses the RID of each record as a key that maps to the corresponding physical location of that record

**Page Range**

Base Page 1 (set of physical pages)

Base Page 2 (set of physical pages)

Each Page Range has a set of Base Pages and Tail Pages associated with it. The number of Base Pages per Page Ranges is determined by the number of Pages per Base Page getting as close to MAX_PAGE_RANGE_SIZE without going over

**Physical Pages**

Page 1

...

Page 10

A set of Physical Pages is created for each Base Page or Tail Page. A Page is created for each column in the table

# Table: Metadata

- RID: 9 digit record ID

1 *locType*

23 *locPRIndex*

45 *locBPIndex*

6789 *locPhyPageIndex*

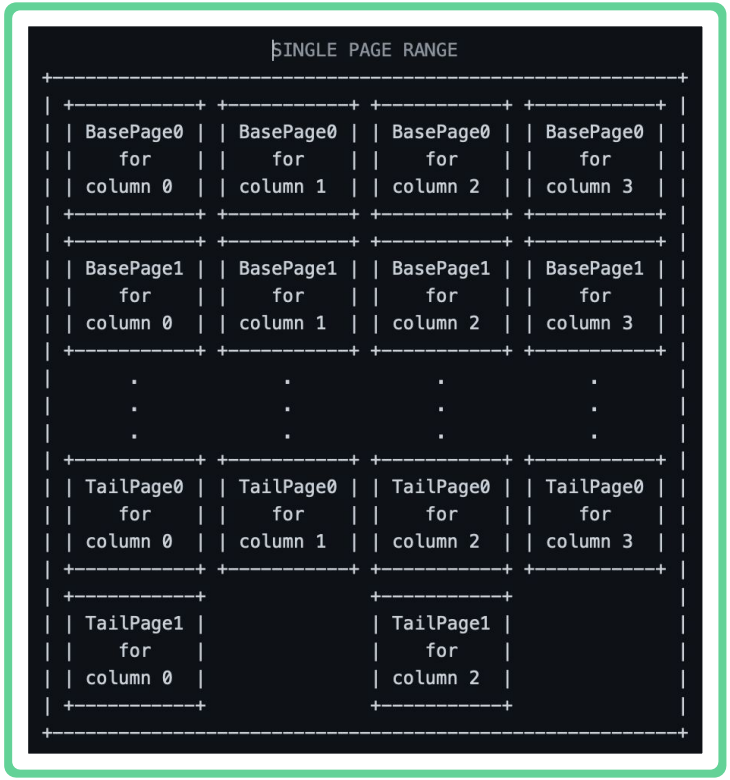
- Timestamp: marks the time of entry
- Schema encoding: integer indicating updates

0 *unchanged*

1 *updated*

2 *deleted*

- Indirection: points to the latest update



# Physical Page

Each Page is 4000 bytes and can store 1000 records. Each integer record is first compressed into Hexadecimal and then inserted into the ByteArray at 4 bytes per record.

**Compress:**
**integer → Hex → Byte**

write(self, value)

replaceRecord(self, value)

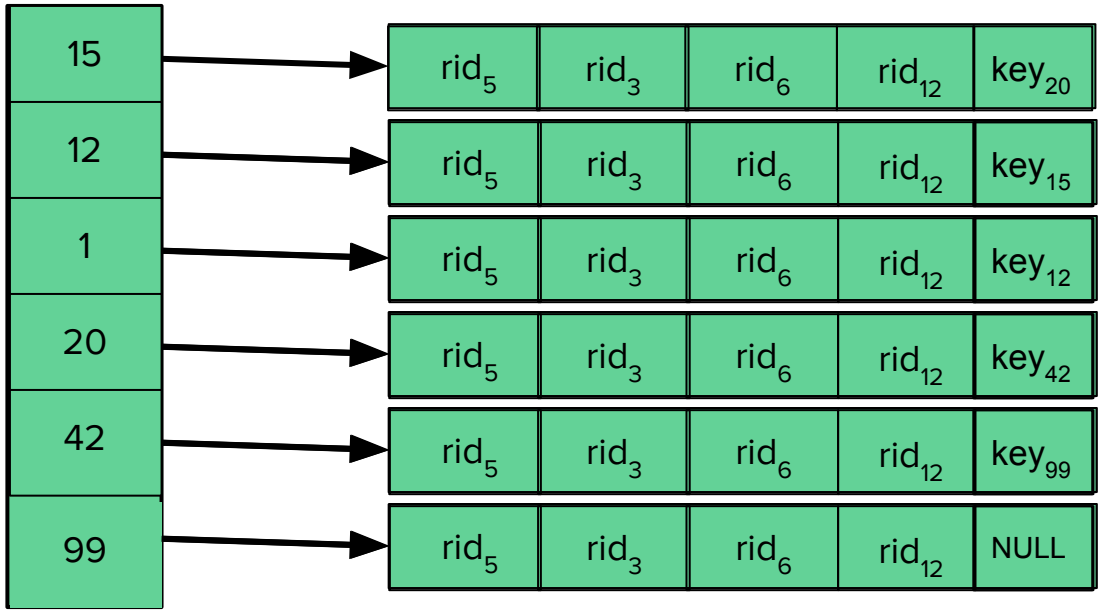
| index | Byte Array |
|---|---|
| [0,3] | b'\x00\x00\x00\x00' |
| [4,7] | b'\x00\x00\x00\x00' |
| [8,11] | b'\x00\x00\x00\x0a' |

**Decompress:**
**Bytes → Hex → Integer**

getRecord(self, value)

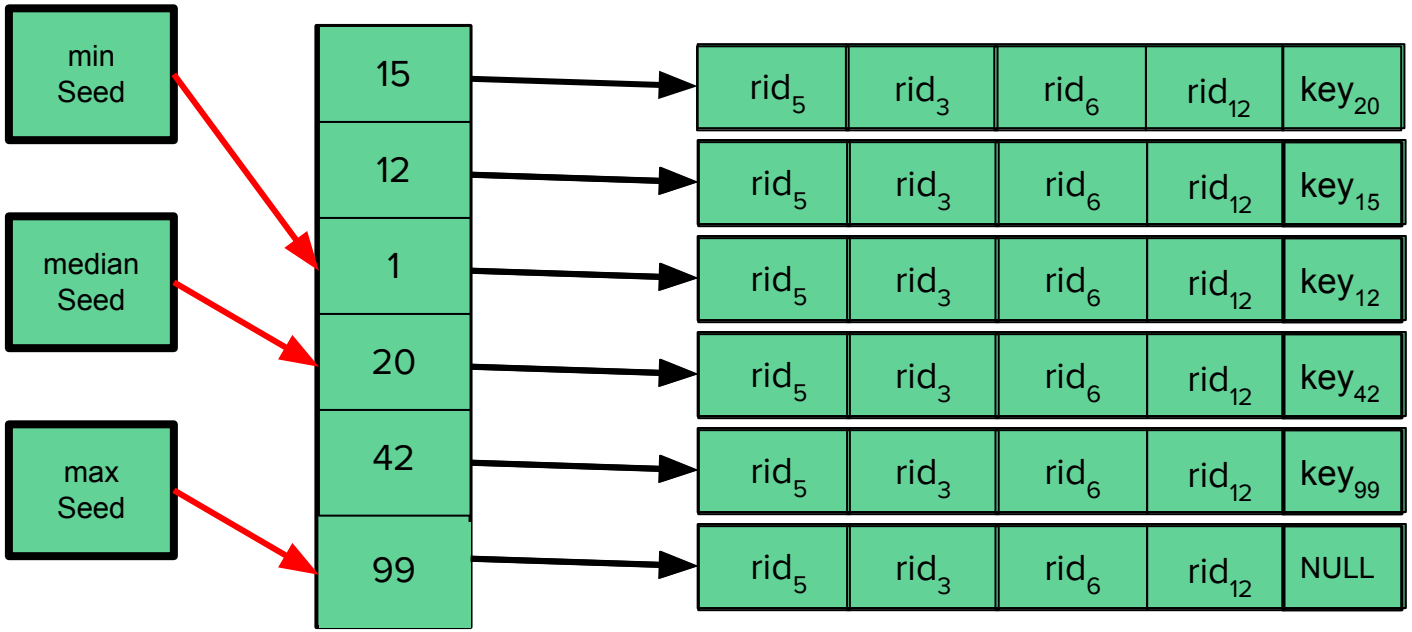
# Index: An RHash Implementation

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



# Index: An RHash Implementation

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



# Query: Methods

## insert(self, *columns)

- Inserts a new record into the base pages with inputted data
- Generates metadata such as timestamp, RID, schema encoding, and indirection for the new record

## update(self, key, columns)

- Creates new records in the tail pages with their indirections pointing to the keyed records
- Changes the keyed records' indirection values to point to the new records

## delete(self, key)

- Creates new records with blank data in the tail pages with their indirections pointing to the keyed record
- Removes the RIDs of the deleted records from the table index

# Query: Methods

## select(self, key, column, query_columns)

- Finds the keyed records through the inputted column
- Returns a list of Record Objects with the requested data based on the inputted bit array

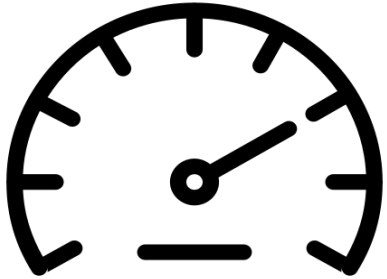
## sum(self, start_range, end_range, aggregate_column_index)

- Searches for the RIDs between the specified range
- Returns the sum of the requested values between the specified range

# Query: Runtimes

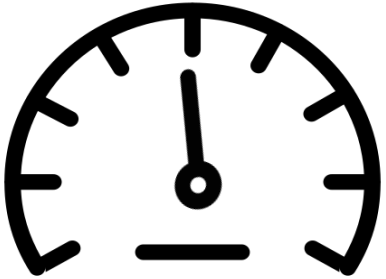
Below are the fastest execution times in seconds for the _main_.py file using 10,000 records:

Insert



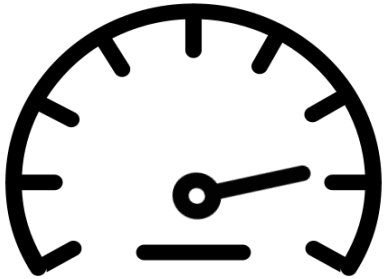
0.543902s

Update



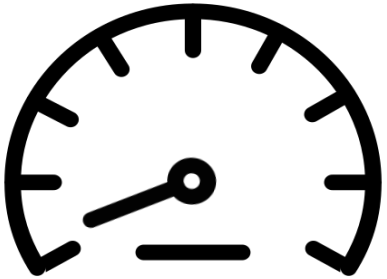
0.919173s

Select



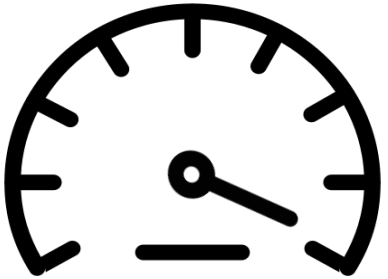
0.290433s

Aggregate



9.933973s

Delete



0.048569s

# Conclusion

Some aspects of the project that we would have liked to improve:

- Decrease run-time by optimizing key methods
- Linked List structure for Base and Tail pages
- Creating more seeds for the RHash Index
- Using bitwise methods and operations to generate RID

*Source: https://www.elegantthemes.com/blog/tips-tricks/10-effective-website-improvements-that-take-10-minutes-or-less-to-do*

# Questions