L-Store
Milestone 1

ECS 165A

Natheenthorn Teachaurangchit
Michael Shaw
Stuart Feng
Henry Chou
Eric Wang

Data Model

Data Storage
Base and Tail Page
Page Range

Bufferpool Management

Page Directory
Index Directory

Query Interface

Select
Update
Sum

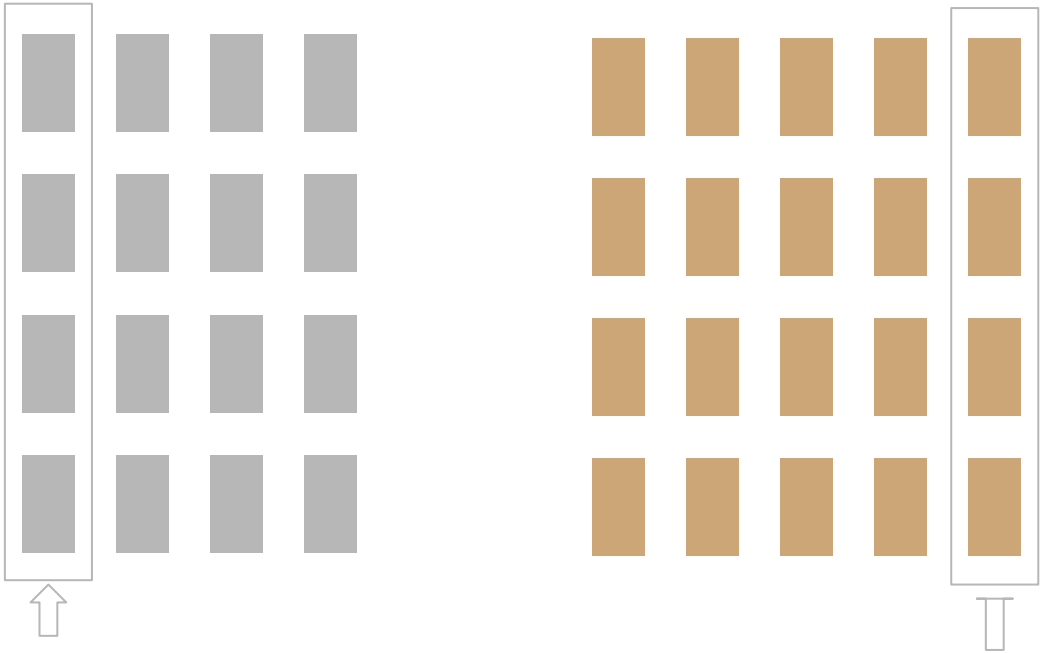
Insert
Delete
Increment

(S1) Data Model

Column-based Data Storage

Metadata (Indirection, RID, Timestamp, schema encoding)

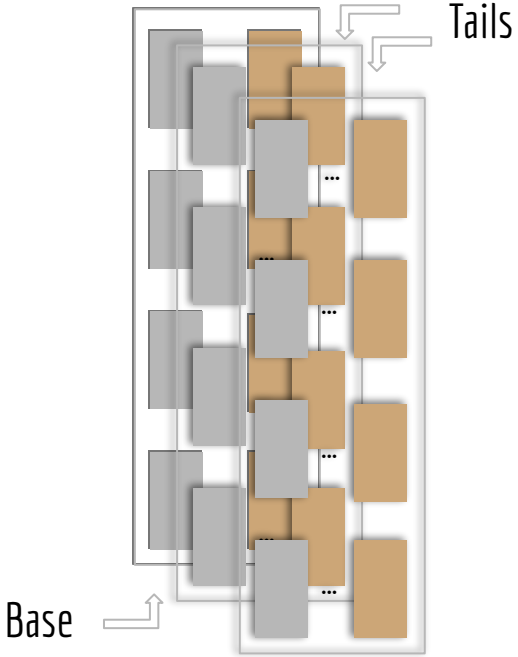
Data Values



Page 1

Page # = num_columns + METADATA_CT (4)

Base and Tail Page



Design decision (Python Objects)

Initially (All the pages were stored in an array):

[
  [[base_page], [base_page]], #Page Range 1
  [[base_page], [base_page]] #Page Range 2
]

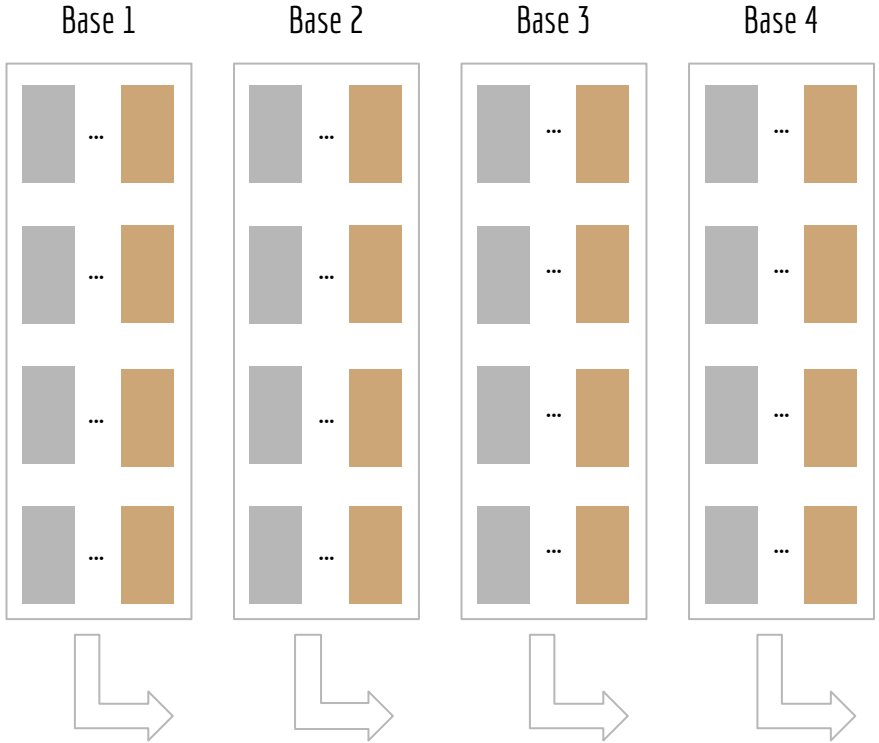
Final Decision (Store everything in an object):

[PageRange(BasePage(), ...)]

Page Size = 4096 bytes, **Record Size** = 8 bytes
Therefore, we could store 512 records per page

Page Range

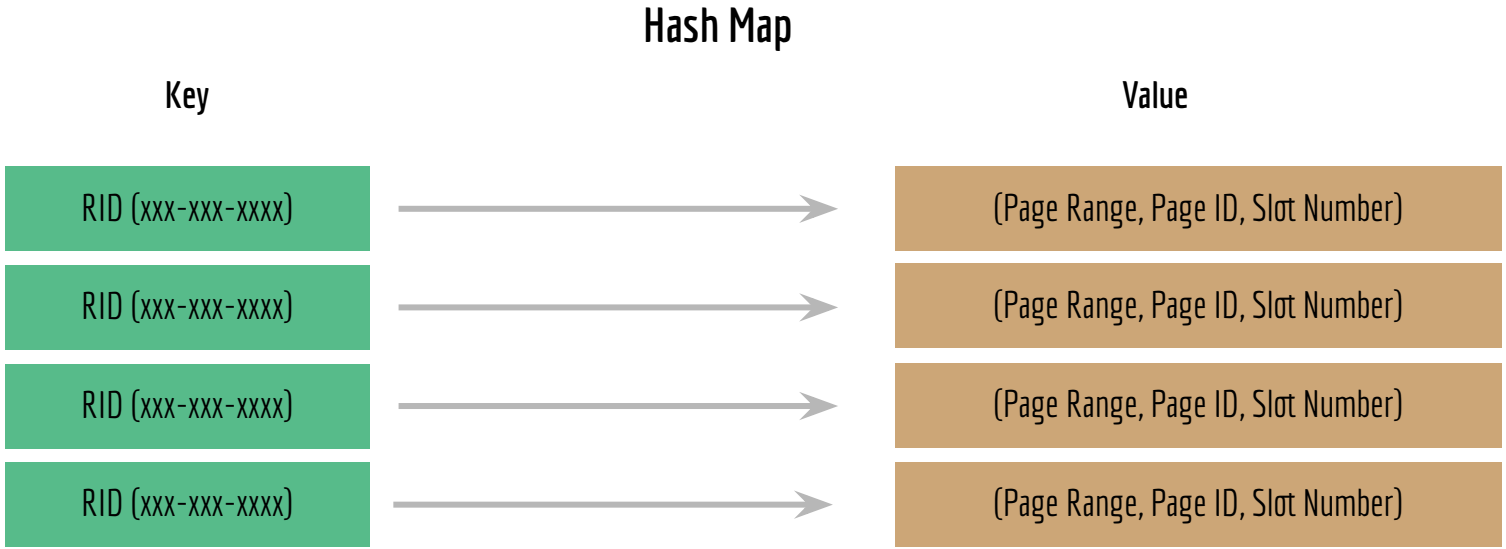
Defaults to 4



Each page range should be able to store 8192 records in the base pages. This should make it easier to ease into the next milestones of dividing the workload across multiple threads without concurrency issues.

(S2) Bufferpool Management

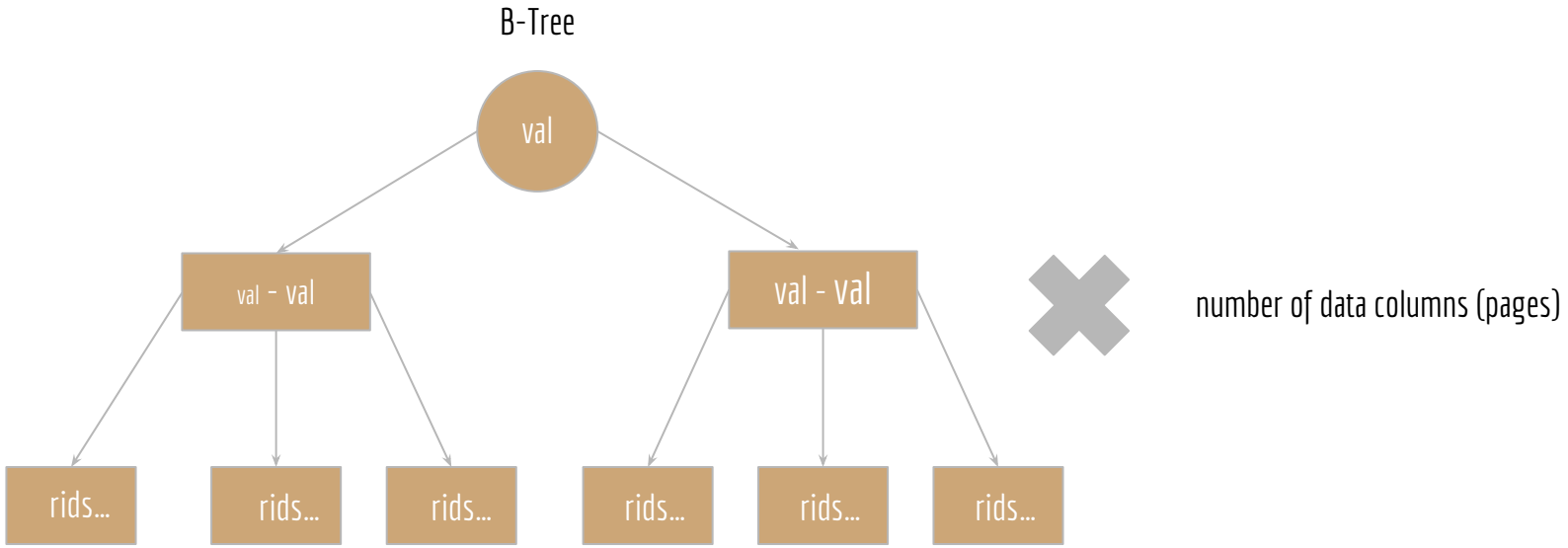
Page Directory



Decision (HashMap) : Since all the RIDs are unique, there should be no collision when hashing; therefore, HashMap would give us the best performance.

Index Directory

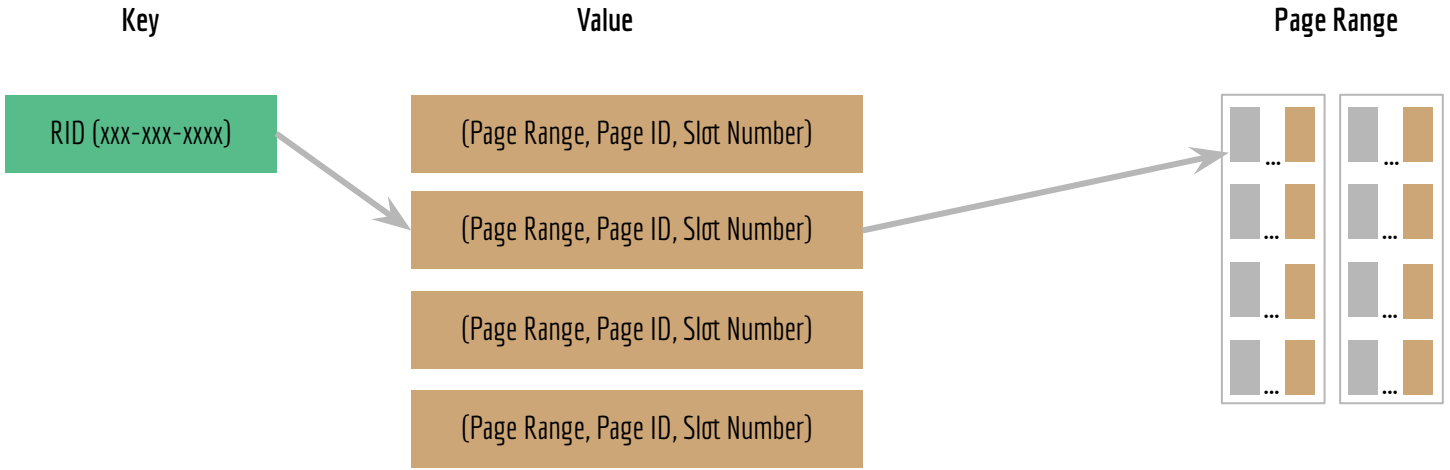
Design Decision (B-Tree): Since we are implementing index for all of the columns, we acknowledge that there will be columns with duplicate values. Therefore, using B-Tree will yield the best performance, while maintaining appropriate memory allocation.



(S3) Query Interface

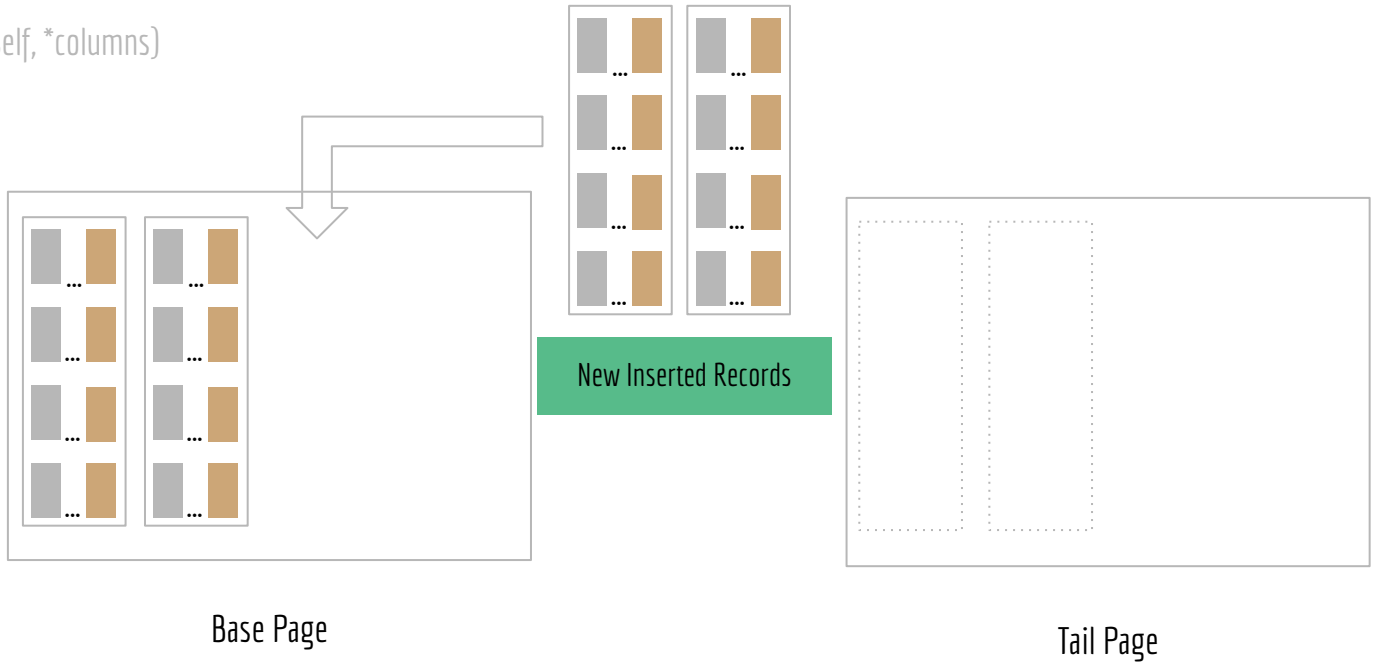
Select

select(self, index_value, index_column, query_columns)



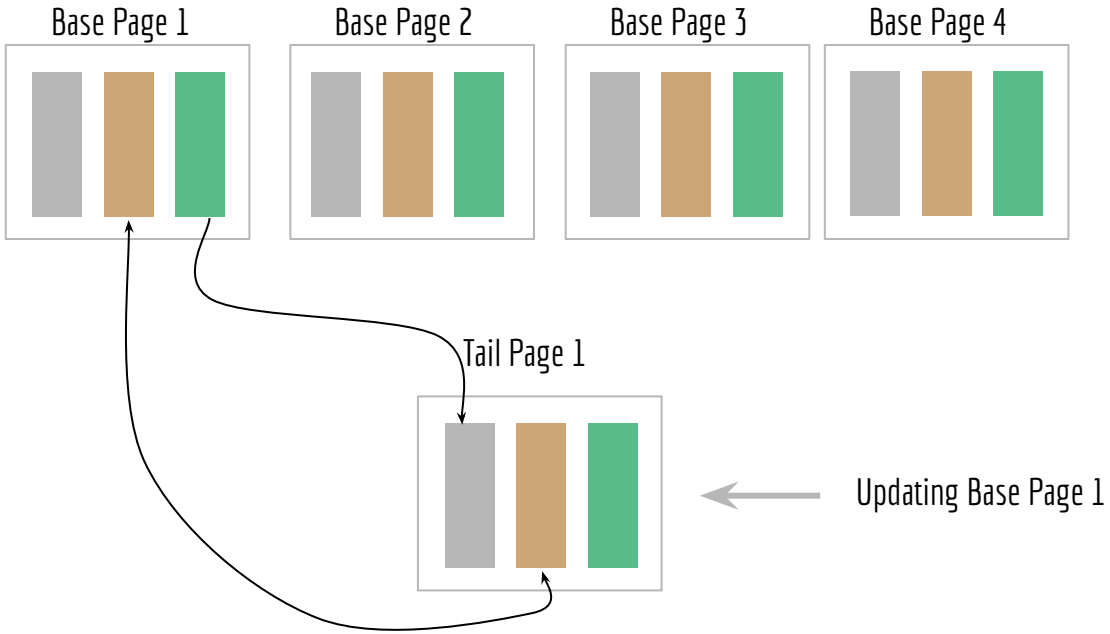
Insert

insert(self, *columns)



Update

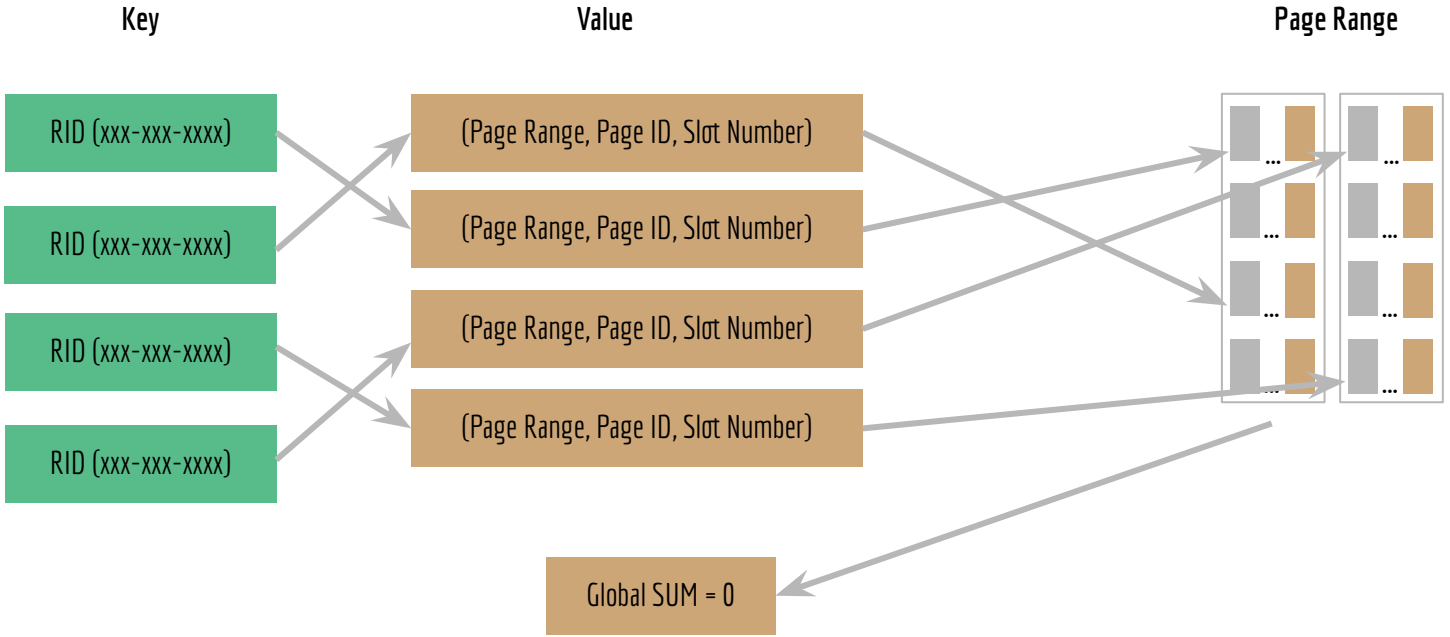
update(self, primary_key, *columns)



Sum

sum(self, start_range, end_range, aggregate_column_index)

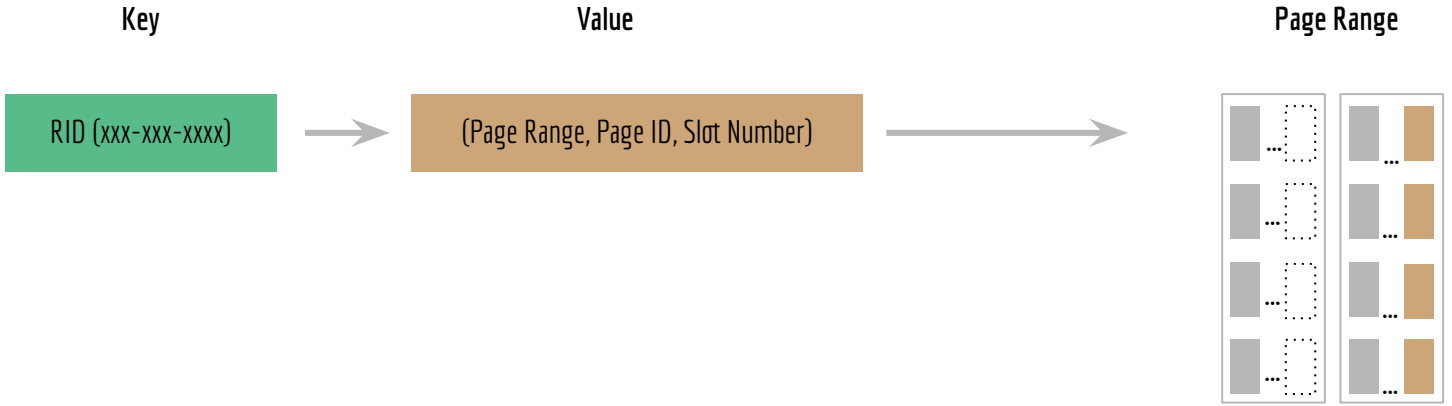
Straight forward check for range of columns and aggregates the sum of their values



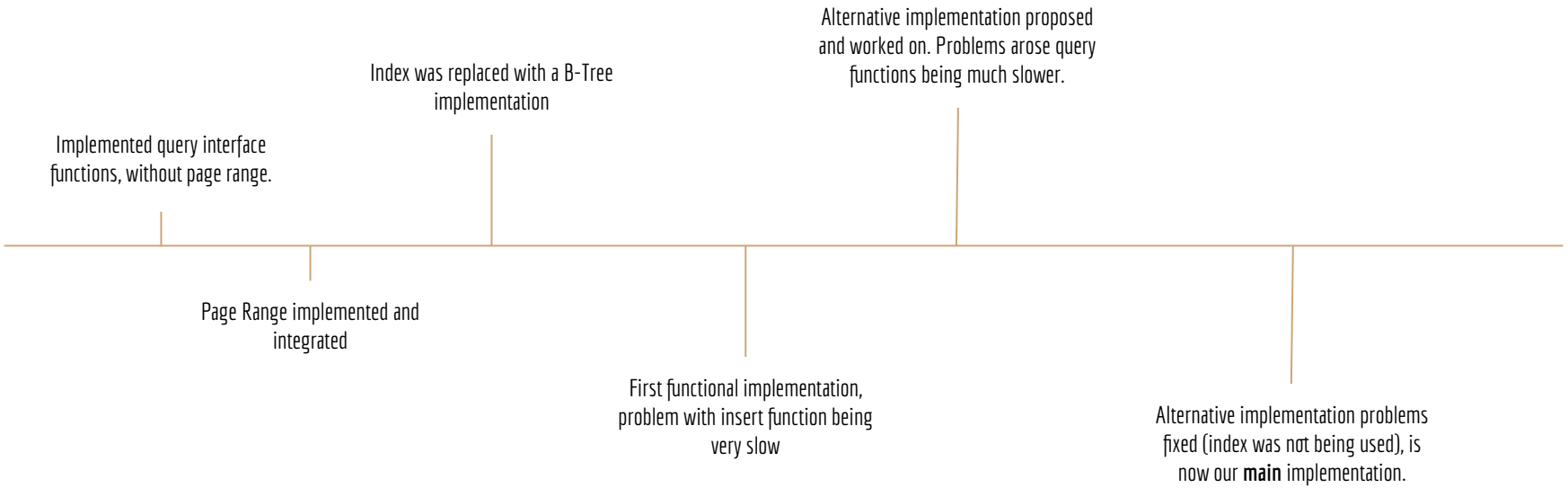
Delete

delete(self, primary_key)

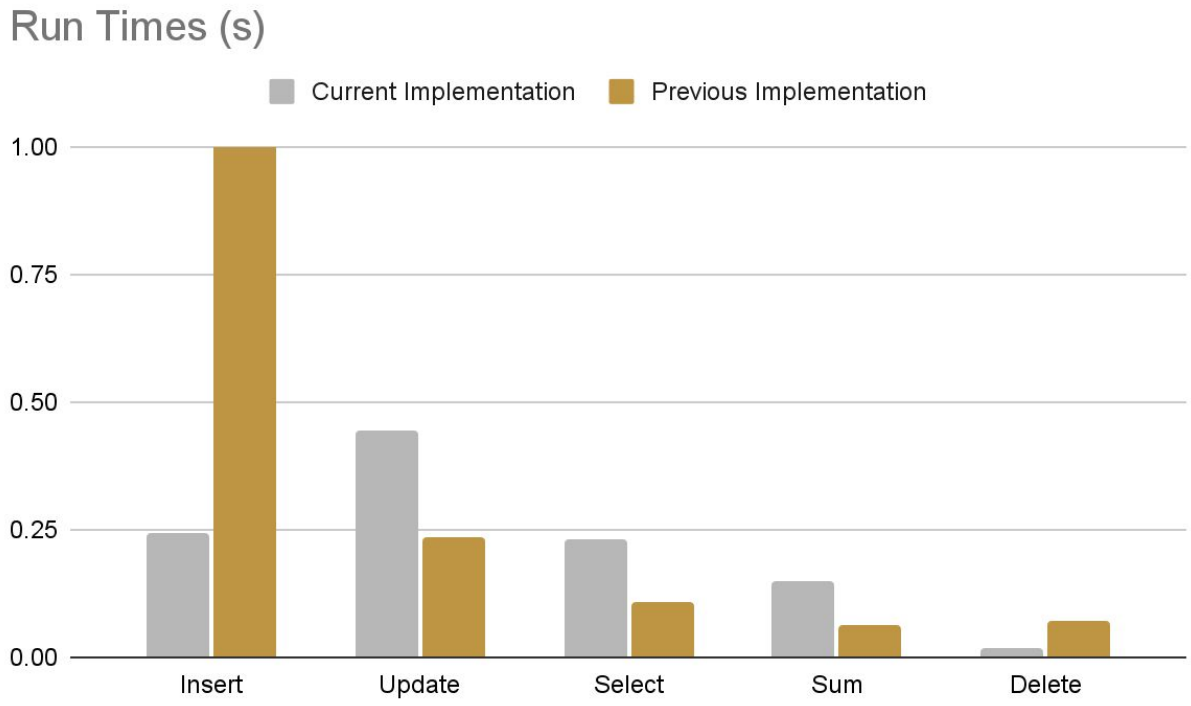
Straight forward deletion method using provided primary key to use equivalent RID and remove target record in base page



Roadmap


Implemented query interface functions, without page range.
Page Range implemented and integrated
Index was replaced with a B-Tree implementation
First functional implementation, problem with insert function being very slow
Alternative implementation proposed and worked on. Problems arose query functions being much slower.
Alternative implementation problems fixed (index was not being used), is now our **main** implementation.

Current & Alternate Implementation Speed Comparison



Thank you!