# LTeam Milestone 1

Alejandro Torres, Jenny Wang,
Ho-Chih Ma, Jamie Wu,
Karthik Palanisamy

# Team Member Roles

Leadership Roles:

- Team Coordinators: Jenny, Alejandro
- System Architects: Everyone
- Developers: Everyone
- Testers: Everyone

Implementation and Design Areas:

- Query Evaluation: Jenny, Alejandro, Karthik, Jamie
- Bufferpool Management: Alejandro, Ho-Chih
- Crash, recovery, logging: N/A
- Synchronization and Concurrency: N/A

# (1) Data Model

Pages
Bagepages and Tailpages
Pageranges
Table

# (2) Bufferpool Management

Page Directory
Index Directory

# (3) Query Interface

Insert
Update
Select
Sum
Delete

# (4) Performance

Btree vs Hash Table vs B+Tree
Pagerange and Page Sizes
Overall Runtime

# (5) Live Demo and Q&A

# [1] Data Model

Pages
Basepages and Tailpages
Pageranges
Table

# Pages

Page 0

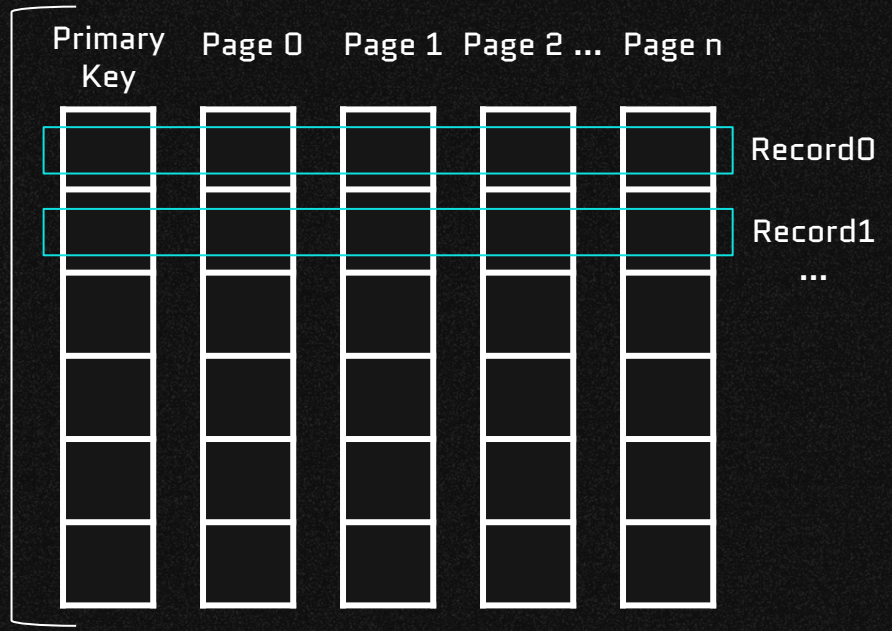

- Each Page is 4096 bytes, which can hold 512 records
- Each Page represents a singular data column
- Choose to use bytearrays instead of traditional arrays for writing to memory in future milestones

# Basepages and Tailpages

Basepage:
- RID = []
- StartTime = []
- SchemaEncoding = []
- Indirection = []
- Pages = [Pages[]...]

Tailpage:
- RID = []
- Indirection = []
- Pages = [Pages[]...]

Pages Array

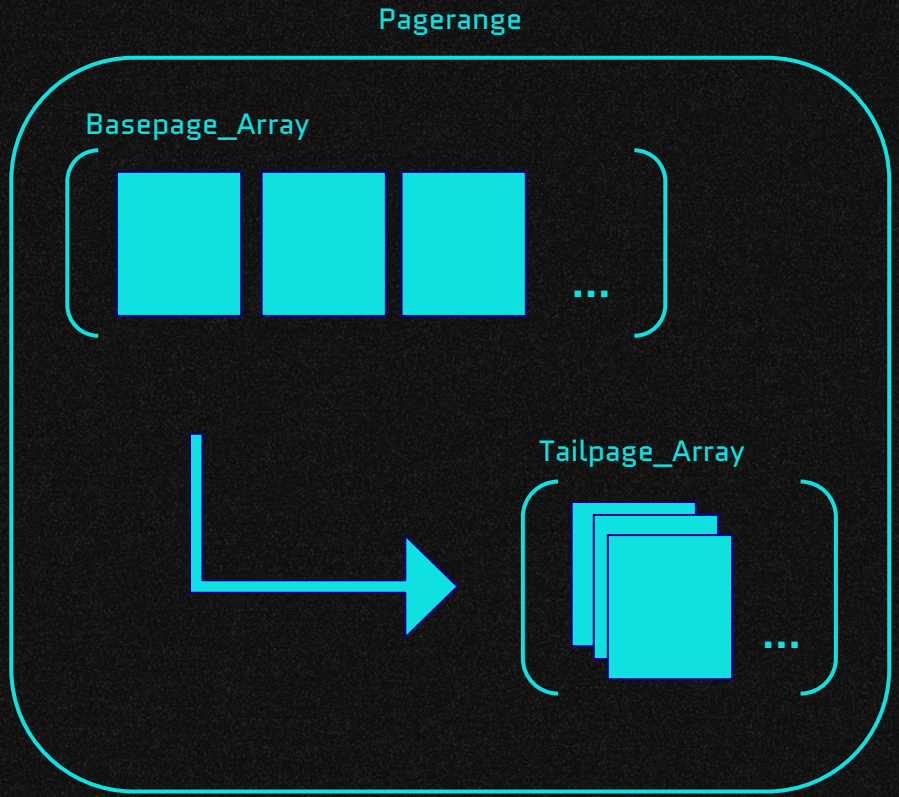



- We make a new Basepage or Tailpage whenever either exceeds 512 records

# Pageranges

Pagerange:
Basepage_Array = [Array of Basepage Objects...]
Tailpage_Array = [Array of Tailpage Objects...]

- Each Pagerange can only hold 16 Basepage Objects
- We ensure Tailpages are the granularity of each pagerange



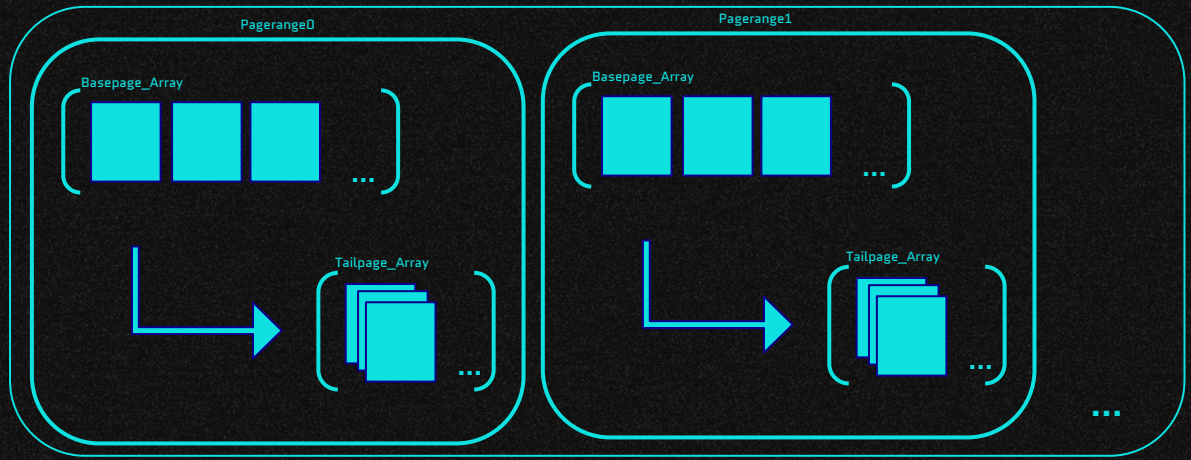
# Table

Table:

- Name
- KeyIndex
- NumColumns
- PageDirectory
- Index
- IndexOfBasepageArray
- IndexOfTailpageArray
- PageRangeArray = [Array of Pagerange[] objects...]

Overall Purpose of our Data Model:

- Easy to keep track of which Index belongs to which Table, which Tailpage belongs to which Pagerange, etc
- Everything that is related is grouped together by classes



# [2] Bufferpool Management

Page Directory
Index Directory

# Page Directory

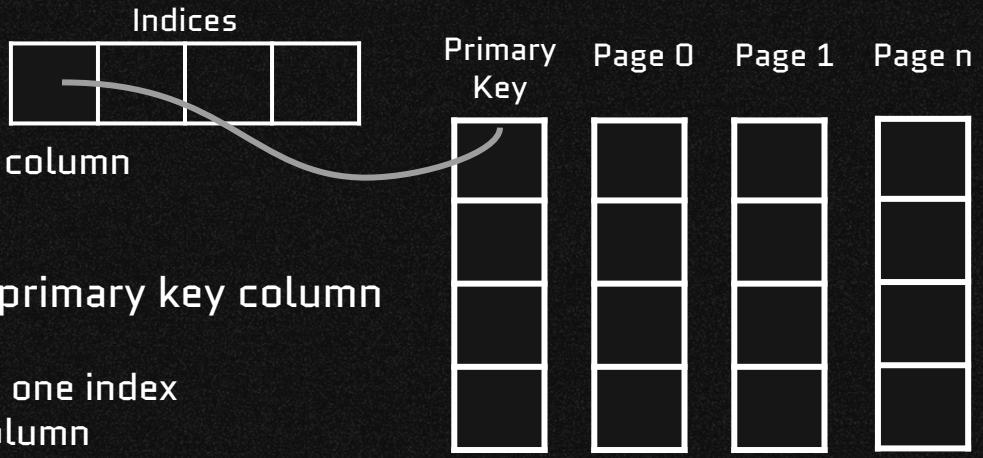
Objective: map base page RIDs to the newest version of the record. (Indirection)

- Page Directory is generated each time a record is inserted and updated each time a record is updated
- We define it as a dictionary, because internally it is implemented as a hash table in python

# Index Class & BTree

Objective: Given a column and its value, return the RIDs associated with

- One Index Per Table
- Indices are BTrees
- Aim to have an index for each column




Limitation: only implemented primary key column

- For now, each table only has one index
- One BTree for primary key column
- Primary key : RID of the Base Page

# [3] Query Interface

Insert
Update
Select
Sum
Delete

# Insert

Objective: Insert new record into the Basepage[] AND maintain LStore fundamentals

Two Checks Required:

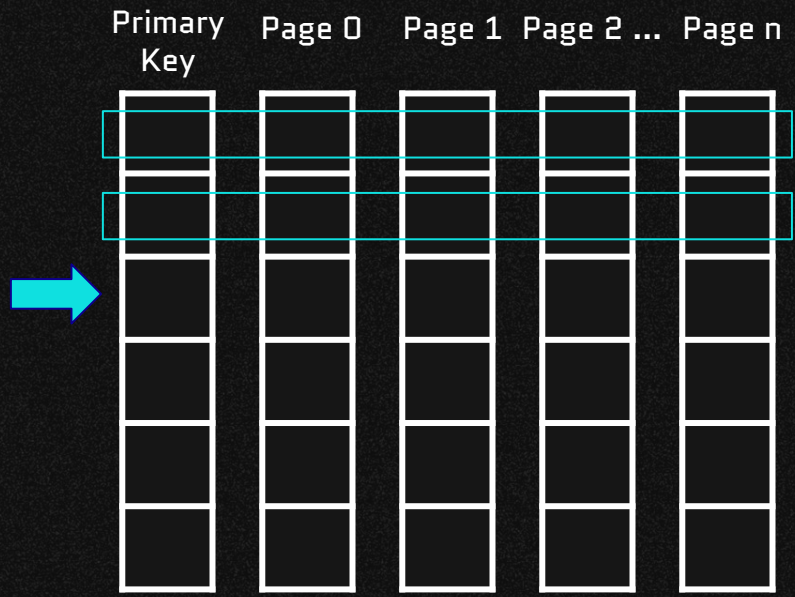
1. If we've hit the max number of records allowed in one page, then we need to make a new Basepage[]
2. If we've hit the max number of Basepages allowed in a pagerange, then we need to make a new Pagerange[]

RID:

- Create record's RID as a tuple (Index in Pagerange, Index in Basepage, Index in Page , 'b')

BTree:

- Insert primary key:RID into a node



# Update

Objective: Update record into the Tailpage[] AND maintain LStore fundamentals

Required Check:

1. If we've hit the max number of records allowed in a Page, then we make a new Tailpage[]

Challenge:

- Updating and maintaining the Indirection column and update lineage

RID:

- Tuple as [Index in Pagerange, Index in Tailpage, Index in Page , 't']

| RID | Indirection [Primary key, Page0, Page1] | |
|---|---|---|
| (0, 1, 24, 'b') | None | [1,1,1] |

Original basepage record
[never updated before]

Update[ 1, [None,2] ]

Basepage record

| | | |
|---|---|---|
| (0, 1, 24, 'b') | (0, 0, 16, 't') | [1,1,1] |

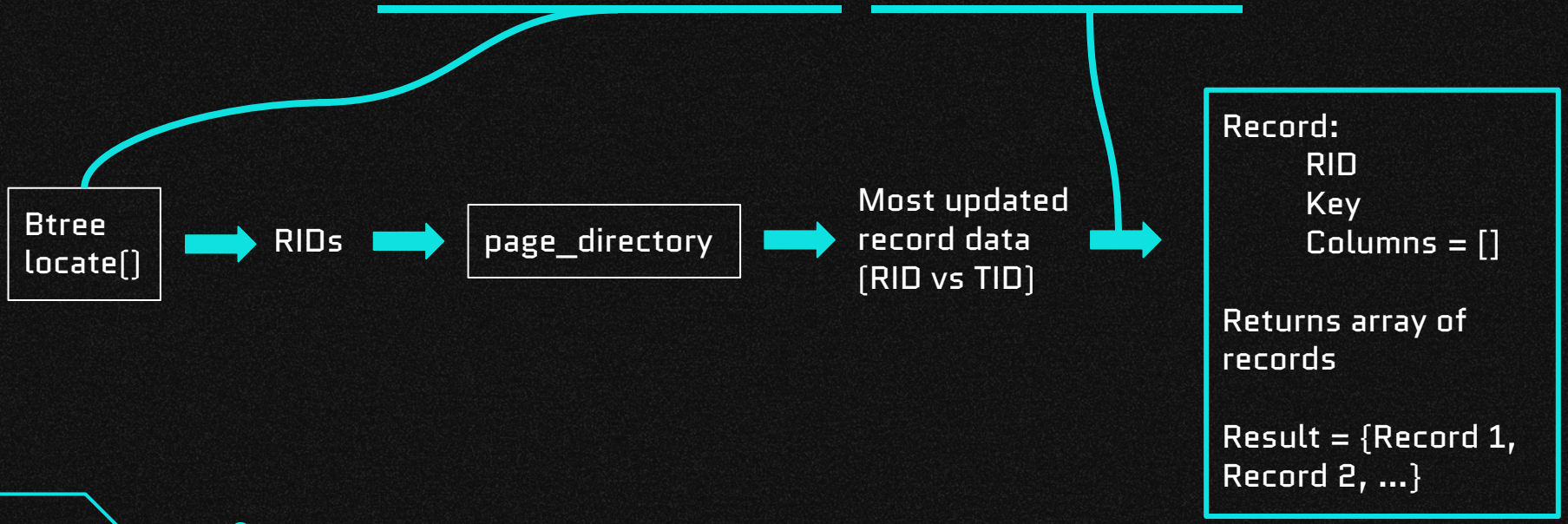
Tailpage record

| | | |
|---|---|---|
| (0, 0, 16, 't') | (0, 1, 24, 'b') | [1,1,2] |

# Select

Objective: Based on one known attribute/condition, look up other column data

Syntax: query.select(search_key, search_key_index, projected_columns_index)



# Sum

Select SUM(key) as total FROM grades_table

Locate_range returns a list of RIDS

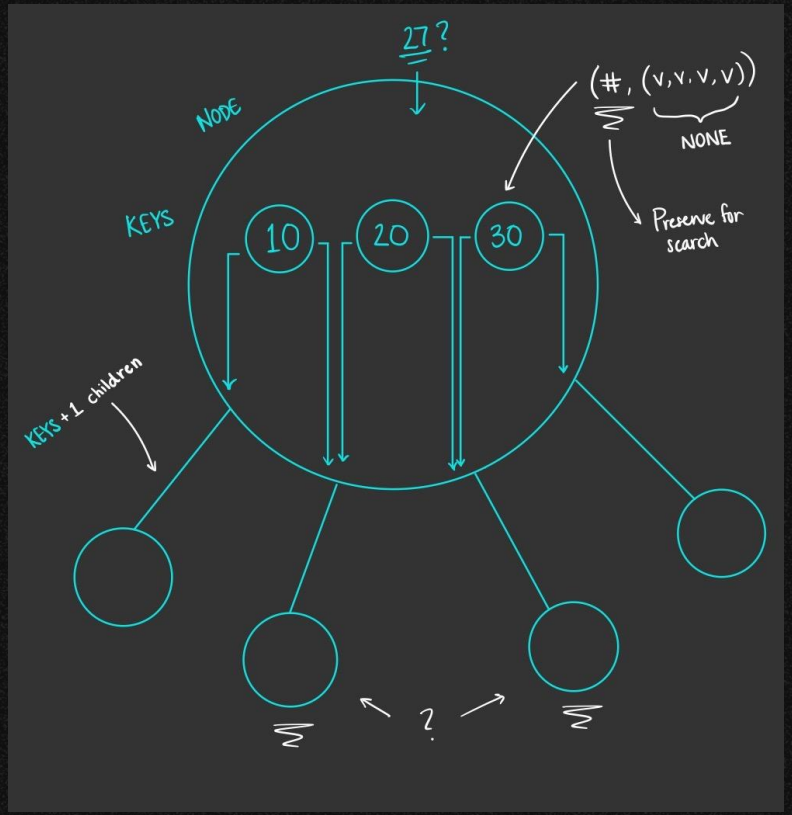
Locate_range

Physical page

total

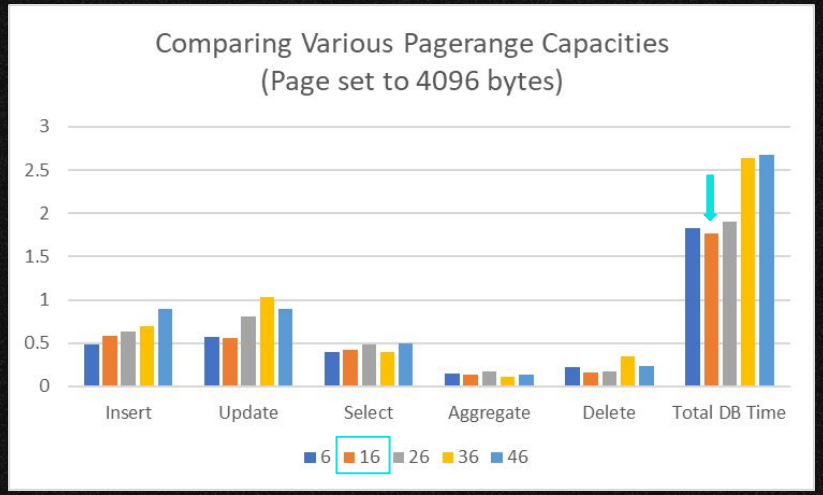
Note: RIDS are all the newest versions of the record

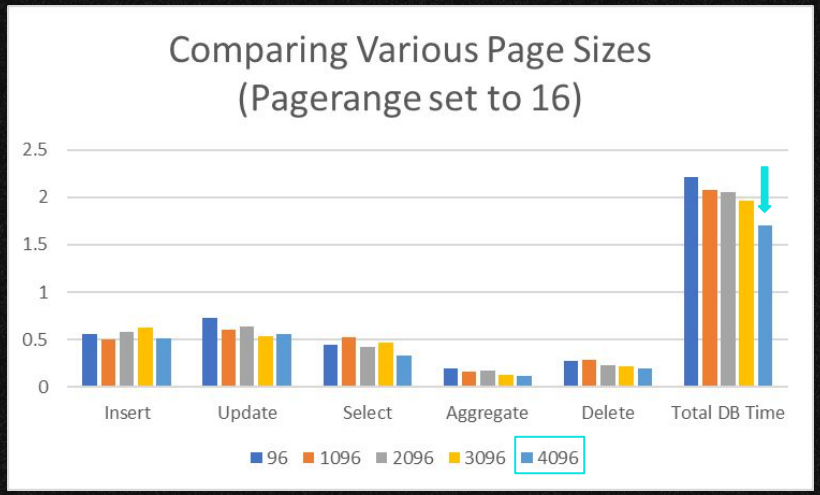
# Delete


27?
NODE
KEYS
10
20
30
(#, (v,v,v,v))
NONE
Preserve for search
KEYS + 1 children
?

# [4] Performance

# Pagerange and Page Sizes




Our final decisions:

- Pagerange Capacity = 16 Basepages
- Page Size = 4096 Bytes (512 Records)

# Overall L-Store Performance

Issues: Select query gave us the slowest results.

- Page directory was initially being generated in the select method (very slow)

```
Inserting 10k records took:                         0.21875
UPDATE----------------------
Updating 10k records took:                          0.21875
Selecting 10k records took:                         0.125
Aggregate 10k of 100 record batch took:  0.0625
Deleting 10k records took:                          0.078125
total db time: 0.703125
```

# (5) Live Demo and Q&A