Milestone 1

Yingchen Gu, Glenn Chen
Rishika Roy, Kaleb Crans,
Ryan Kim
Overview

Data Model
- Database Structure
- Table: Page Range
- Base and Tail Page
- Physical Pages

Bufferpool Management
- Table: Page Directory
- Indexing

Query Interface
- Select
- Insert
- Update
- Delete
Data Model
Database Structure

Table

Page Directory

Page Range 1

Page Range 2

... 

Up to 16 Base Pages

As many Tail Pages as needed

Metadata

Column Data

Metadata

Column Data
**Base Page and Tail Page**

Base Page: 4096 kb
8 bytes each

**Tail Page**

<table>
<thead>
<tr>
<th>IND</th>
<th>RID</th>
<th>SCH</th>
<th>TIME</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Page: 512 records per base page**

<table>
<thead>
<tr>
<th>IND</th>
<th>RID</th>
<th>SCH</th>
<th>TIME</th>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Physical Pages

Base Page
(Total 16 Base Page)
Has its own set of physical pages
A physical page for each column
4kb per physical page

Tail Page
Similar to the base page, the tail page has many physical pages

Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Physical Page
Table.py: Insert

If the latest page range has space, add a base record to the base pages.

Also add a tail record to the tail pages of the same page range.

This is the initial tail record for the base record, and prevents issues with indirection and merging for later milestones.
If the Page Range is full (has 16 base pages in them already), then a new page range would be added.
Table.py: Update

Add a tail record to the same page range that the base record that is being updated is in

Update the base record’s indirection to point to the new tail record

Make the new tail record’s indirection point to the previous latest tail record

Cumulative tail records - each tail record contains the latest updated values
def has_capacity(self):
    # Checks if there are space open for new write functions
    # If 512 records (4096kb) are reached, create a new page

def write(self, value, index):
    # The value will be placed in the specified column of the index
    # It will increment each space by 8 bytes so that every number will take a total of 8 bytes
Bufferpool Management
Stores key-value pairs of RIDs and tuples that contain a location in the table

- Which page range
- Base or tail page
- Which physical page
- Offset in physical page

Key-value pairs are created whenever a record is added
Indexing

Multiple dictionaries (key-value pairs)
- one for each data column
- data values stored as keys
- RIDS stored as values

Provides `locate` and `locate_range` functions to return RIDSs for any data value or range of data values for a specified column.
Query Interface
def insert(self, *columns)
    self.table.insert(*columns)

def update(self, *columns)
    self.table.update(*columns)

Both the insert and update functions in Query uses the insert and update function from Table
Select

Ex:
Select Name column for all with age 19
1. Find RID of all that matches ‘age 19’
2. Get name from those RIDs

<table>
<thead>
<tr>
<th>RID</th>
<th>NAME</th>
<th>AGE</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>13414</td>
<td>Bob</td>
<td>19</td>
<td>97%</td>
</tr>
<tr>
<td>13415</td>
<td>Mary</td>
<td>23</td>
<td>72%</td>
</tr>
<tr>
<td>13416</td>
<td>Josh</td>
<td>20</td>
<td>89%</td>
</tr>
<tr>
<td>13417</td>
<td>Grace</td>
<td>19</td>
<td>94%</td>
</tr>
<tr>
<td>13418</td>
<td>Sabrina</td>
<td>19</td>
<td>73%</td>
</tr>
</tbody>
</table>

Purpose: show desired records
Input
○ index_value, index_column, query_columns
Output
○ Returns a list of record objects upon success
○ False if record locked by 2PL

Returns:
Bob, Grace, Sabrina
Purpose: sum record values in a specified columns

Input
- start_range, end_range,
- Aggregate_column_index

Output
- Sum of given range upon success
- False if no records exist in given range

Assume:
Choose people with the name starting with H to N

Add the age column

Add Mary and Josh’s age together \( \rightarrow 23 + 20 = 43 \)

Output: 43
Delete

- Purpose: delete record given primary key
- Input
  - Primary key
- Output
  - True upon successful deletion

```python
def delete(self, primary_key):
    # Find the RID location
    # Check to see if it's a Base page or Tail page
    # Change the RID to a special value (DELETED_RID_VALUE) using the write function
```
## Delete: with Tail Page

<table>
<thead>
<tr>
<th>RID</th>
<th>NAME</th>
<th>AGE</th>
<th>GRADE</th>
<th>INDIRECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>13414</td>
<td>Bob</td>
<td>19</td>
<td>97%</td>
<td></td>
</tr>
<tr>
<td>13415</td>
<td>Mary</td>
<td>23</td>
<td>72%</td>
<td></td>
</tr>
</tbody>
</table>

**Base Records**

**Tail Record**

RID: 15632
RID: 17482

Check if it has Tail Page

Change Tail Page RID to **DELETED_RID_VALUE**
Questions?
Thank You!