Milestone One

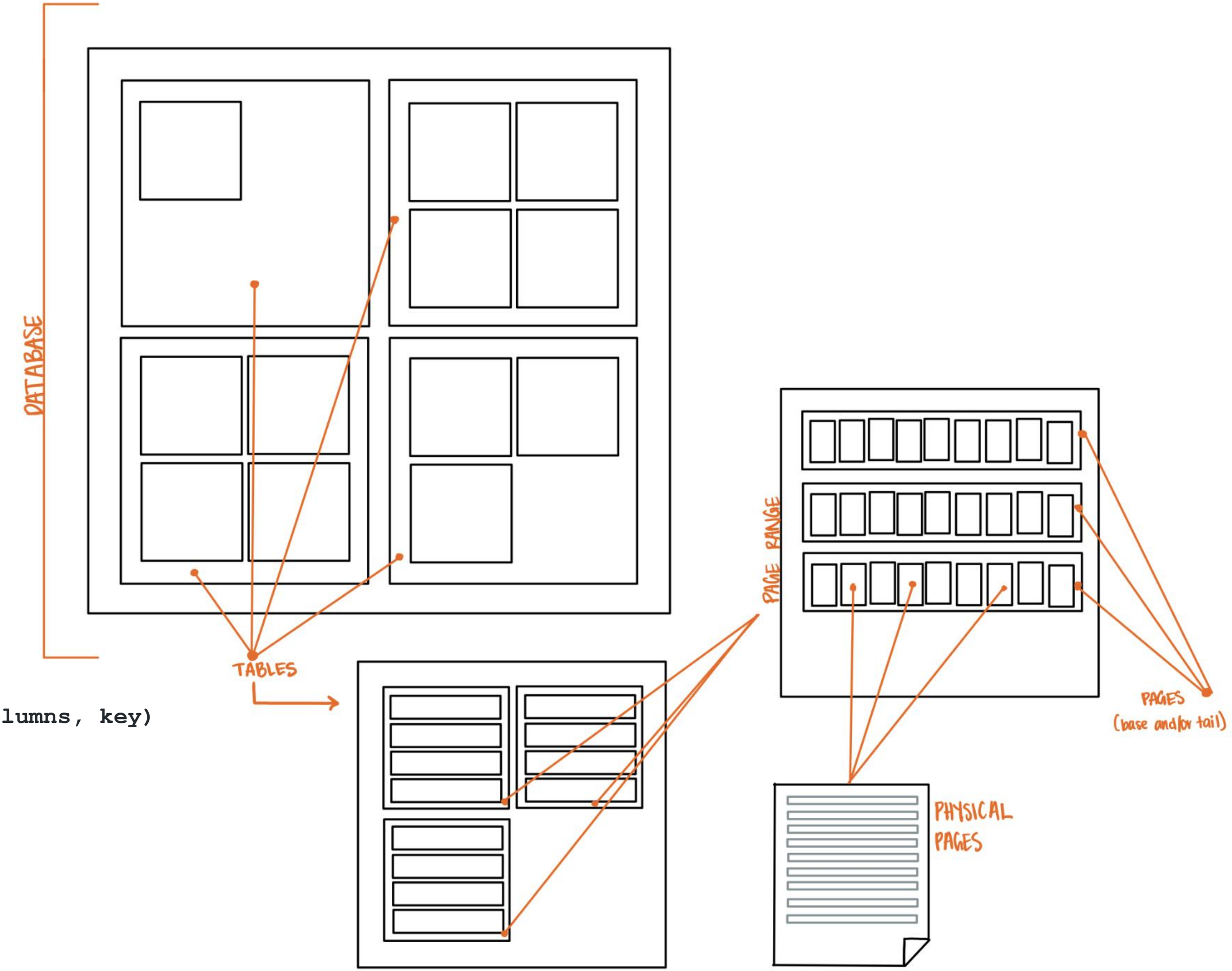
Nick Abcarius
Andrew Do
Travis Garcia
Nicole Pavlovich
Steven Tan

UC Davis, ECS 165A
Prof Mohammad Sadoghi

Milestone Goals

- Create a table and store it in a columnar format
- Design a lineage-based updating solution
- Implement the select, insert, update, delete, and sum queries

Overview



```
class Database():
   def __init__(self):
       self.tables = []
   def create_table(self, name, num_columns, key)
   def drop_table(self, name)
   def get_table(self, name)
```

Layered Design

Table.py	Maps keys to RIDs, sends queries to correct page range, and then calls PageRange.py query functions
PageRange.py	Determines page and physical location for a given query and uses Page.py functions to properly set up records across pages and perform operations on them
Page.py	Keeps track of physical pages and calls the PhysicalPage.py functions that append, read, or update data
PhysicalPage.py	Stores data in a byte array and provides logic to append, read, and update data

Notes:

- Cumulative Update
- Direct physical RID mapping
- Page_directory currently only used to store page ranges

Design Breakdown

```
class Query:

   def __init__(self, table):
       self.table = table


   def delete(self, key)
   def insert(self, *columns)
   def select(self, key, column, query_columns)
   def update(self, key, *columns)
   def sum(self, start_range, end_range,
aggregate_column_index)
   def increment(self, key, column)
```

- Query object performs queries on the data
- Failed queries return false
- Each layer of our design handles different aspects of a query

What the User thinks of:

	Data A	Data B	Data C
1	x	x	x
2	x	x	x
3	x	x	x

```
class Table:
    def __init__(self, name, num_columns, key):
        self.name = name
        self.key = key
        self.num_columns = num_columns
        self.page_directory = [PageRange()]
        self.keyToRID = {}
        self.baseRID = -1
        self.index = Index(self)

    def insert(self, record)
    def update(self, key, record)
    def select(self, key, column, query_columns)
    def delete(self, key)
    def sum(self, start_range, end_range, aggregate_column_index)
    def getPageRange(self, baseRID)
```

self.name, self.key

Entry	Column 1	Column 2	Column 3
1	x	x	x
2	x	x	x
3	x	x	x

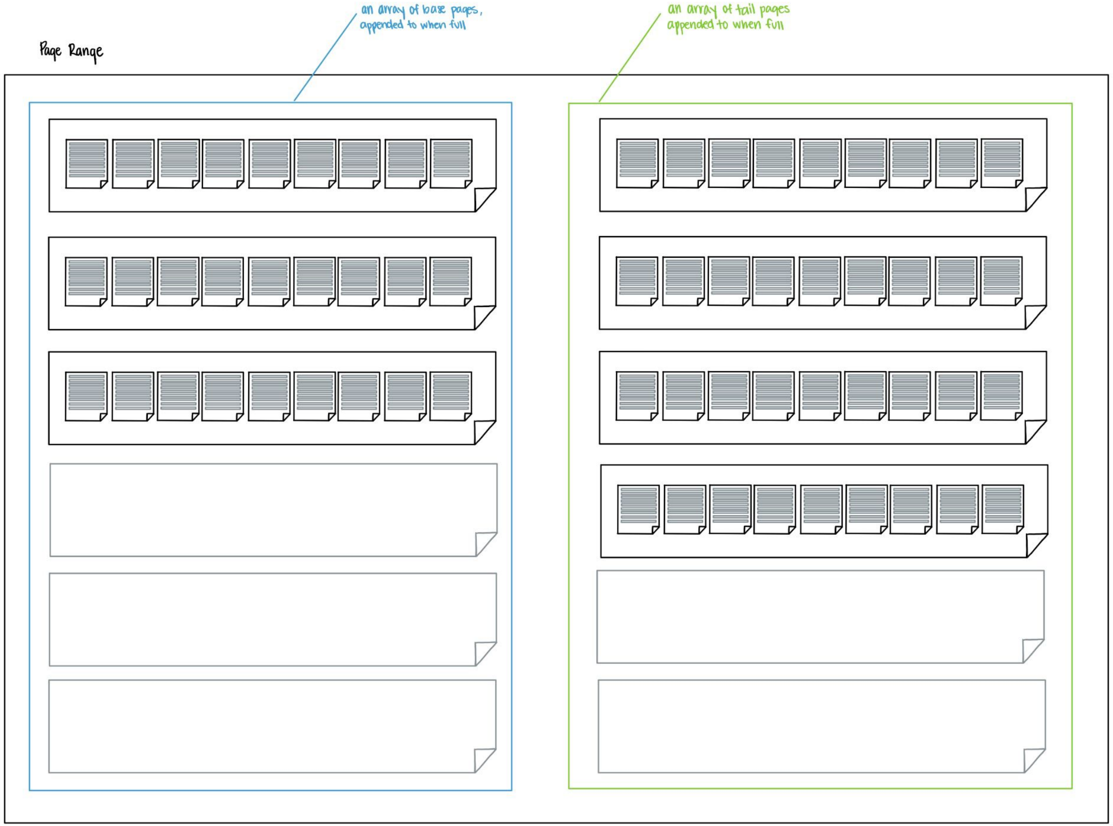
4 columns

Column Breakdown

- **Indirection:** Base indirection points to most updated tail RID. Tail indirections point to previously updated RID
- **RID:** Unique identifier for base and tail records which physically maps to their page locations
- **Timestamp:** Time of last edit/creation
- **Schema**: A single 0 or 1 for our cumulative update which indicates if a record was updated or not
- **Key**: Provided on insert and mapped to a record base RID

Indirection	RID	TimeStamp	Schema Encoding	columns[0] (key)	columns[1]	...

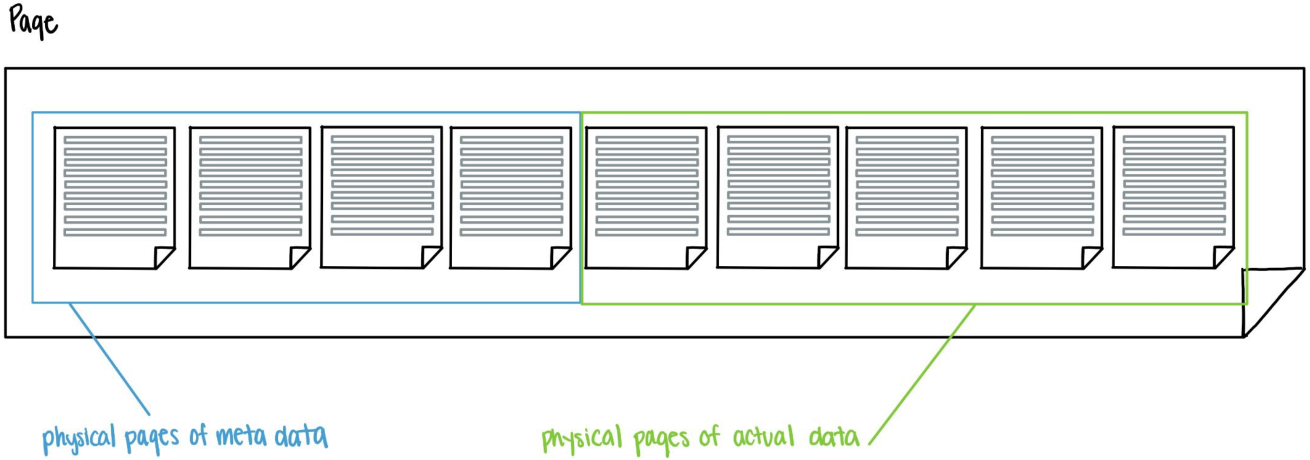
```
class Record:
    def __init__(self, rid, key, columns):
        self.rid = rid
        self.key = key
        self.columns = columns
```



Page Range
an array of base pages, appended to when full
an array of tail pages appended to when full

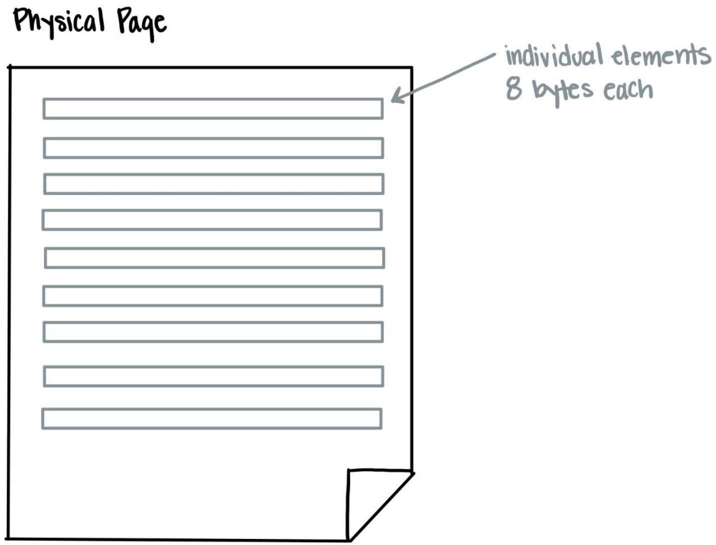
```
class PageRange:
    def __init__(self):
        self.basePages = []
        self.tailPages = []
        self.tailRID = -1

    def baseInsert(self, RID, recordData)
    def tailInsert(self, RID, fullRecord)
    def update(self, baseRID, updatedRecord)
    def getPreviousTailRecord(self,
baseIndirectionRID)
    def select(self, key, baseRID)
    def delete(self, key, baseRID)
    def invalidateTailRecords(self,
indirectionRID, baseIndirectionRID)
    def calculatePageIndex(self, RID)
    def calculatePageOffset(self, RID)
    def addPage(self, record)
    def spliceRecord(self, oldRecord,
updatedRecord)
```



```
class Page:
    def __init__(self, num_columns):
        self.metaColumns = []
        for i in range(0, MetaElements):
            self.metaColumns.append(PhysicalPage())
        self.dataColumns = []
        for columns in range(0, num_columns):
            self.dataColumns.append(PhysicalPage())
```

```
def baseInsert(self, RID, record)
def tailInsert(self, RID, record)
def getRecord(self, offset)
def newRecordAppended(self, RID, pageOffset)
def isFull(self)
def initializeRecordMetaData(self, baseRID)
def invalidateRecord(self, pageOffset)
```



```python
class PhysicalPage:


    def __init__(self):
        self.num_records = 0
        self.data = bytearray()

    def has_capacity(self)
    def appendData(self, value)
    def read(self, location)
    def update(self, value, location)
```

Milestone One

Nick Abcarius
Andrew Do
Travis Garcia
Nicole Pavlovich
Steven Tan

Next Steps:

- Implement indexing and proper page directory
- Look to further optimize our solution
- Begin Milestone 2 requirements