



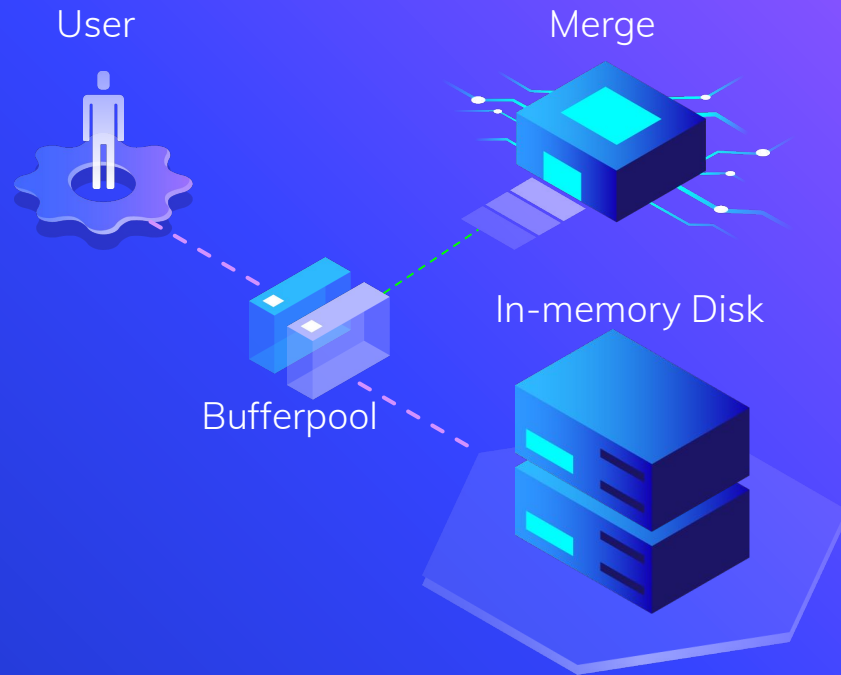
Milestone 2

by The B Team



Big Changes

Durability and Bufferpool Data Management



Architectural Changes

Re-done from milestone 1 for efficiency



Simplified Conceptual Page

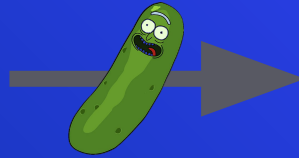
- Each conceptual page (CP) contains a set of columns
1 physical page long

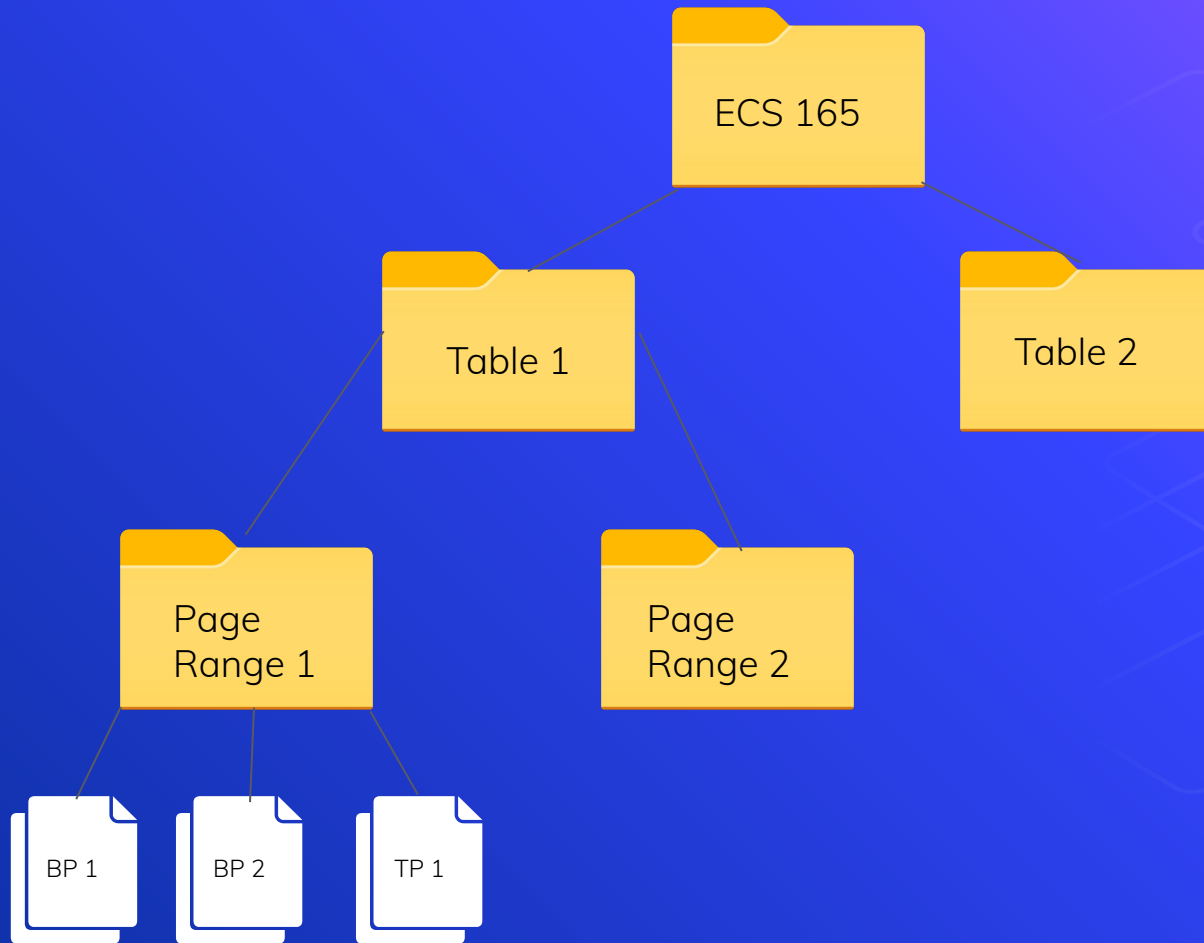


Writing To Disk



- Write CP object to a file using pickle module





Data Management - Bufferpool

- ⬡ MetaData
- ⬡ Conceptual Page granularity
- ⬡ Manages interactions between Query and Disk



Bufferpool - Metadata

- ⬡ Maintains key_dict - > Key : Path
- ⬡ Stores any other dict based on what is indexed
- ⬡ Keeps track of # PR, BP, RID etc.



Bufferpool- Evict

- ⬡ LRU Eviction Policy
- ⬡ Only evicts unpinned pages
- ⬡ Cleans up by locally stored keys

```
### Assuming pages stack will be LRU at top of stack (index 0)
def evict(self): #evict a physical page from bufferpool (LRU)
    ### check if value being evicted is pinned
    # Write to disk whatever is at the top of stack
```

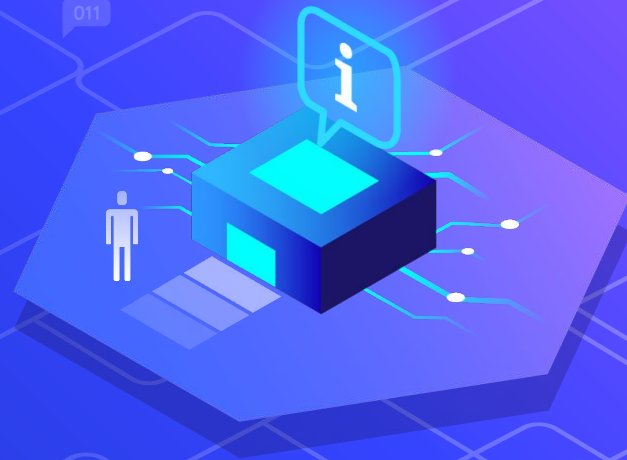
```
    i = 0
    temp_cpage = self.conceptual_pages[i]
    while temp_cpage.isPinned:
        i += 1
        if i == len(self.conceptual_pages):
            i = 0
        temp_cpage = self.conceptual_pages[i]
    self.conceptual_pages.pop(i)
```

```
    self.remove_keys(temp_cpage)
    with open(temp_cpage.path, 'wb') as db_file:
        pickle.dump(temp_cpage, db_file)
    # TypeError: file must have a 'write' attribute
```

```
def remove_keys(self, conceptual_page):
    del self.buffer_keys[conceptual_page.path]
```

Reformatting Query

Including `bufferpool`



New Delete

- ⬡ If not updated before “delete” by adding a tail page with none in columns
 - ⬡ Otherwise: base_schema->-0
- Update page with empty Columns
- ⬡ Update uses bufferpool

```
# If not updated, add tail page with None
if not updated:
    # Update to add tail page with None
    self.update(key, *[None]*n_cols)
else:
    # Change base schema to all 0's, then
    base_schema = np.zeros(n_cols)
    self.update(key, *[None]*n_cols)

return True
```

```
def update(self, key, *columns):
    # """--New--"""
    columns_to_update = self.colsToUpdate(key, *columns)
    my_base_page = self.get_from_disk(key=key)
    tail_page = self.get_tail_page(my_base_page, key, *columns)
    return self.update_tail_page(my_base_page, tail_page, key, *columns)
```

New Insert

- ⬡ Restructured to use bufferpool
- ⬡ Uses the bufferpool to perform insertions

Insert

True

False

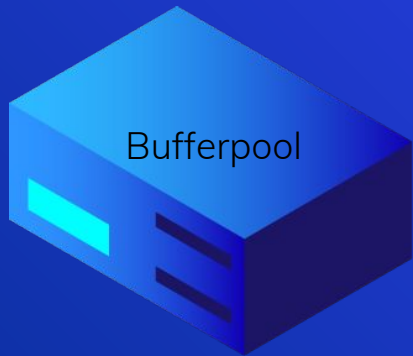
Bufferpool

Disk

```
if new_base_page:
    self.buffer_pool.meta_data.currbp += 1
    path = './ECS165/' + self.table.name + '/PR' + str(self.buffer_pool.meta_data.currpr) + '/BP' + str(self.buffer_pool.meta_data.currbp)
    cpage = self.buffer_pool.createConceptualPage(path, columns)
else:
    path = './ECS165/' + self.table.name + '/PR' + str(self.buffer_pool.meta_data.currpr) + '/BP' + str(self.buffer_pool.meta_data.currbp)
    cpage, is_in_buffer = self.in_buffer(path)
    if not is_in_buffer:
```

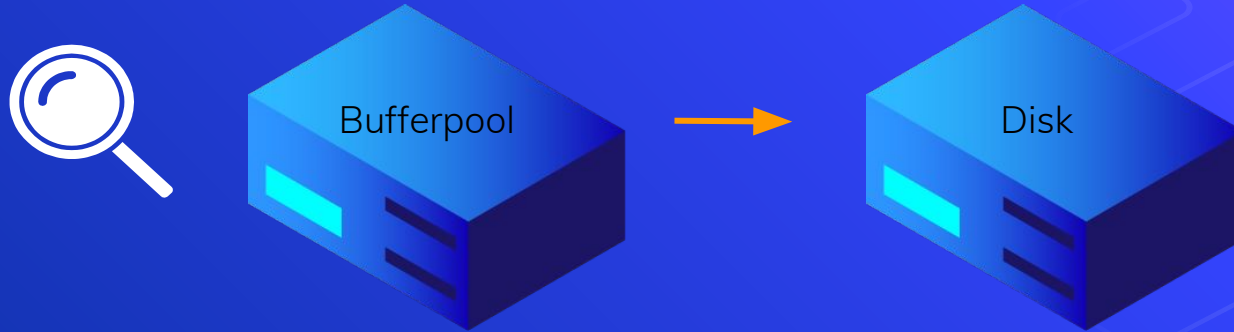
New Select

- ⬡ If we are selecting based on keys simply use `key_dict` and same logic as before
- ⬡ Otherwise we have to utilize secondary indexes if available



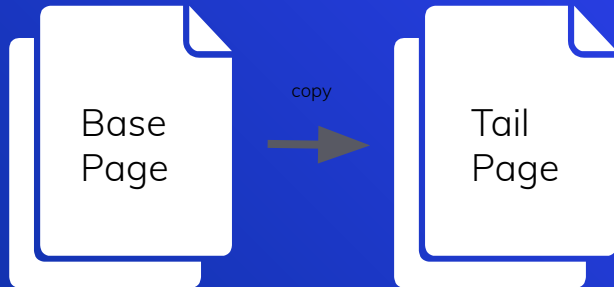
New Update

- Fetch the BP from key
- Find or create the TP associated with given key
- Then update like usual



New Update - Snapshot

- Snapshot upon first update to a specific column in BP
- Copy all query columns from BP into TP



Base Page : [45, 21, 32 , 1, 9]

Update: [17, None, None, 2, None]

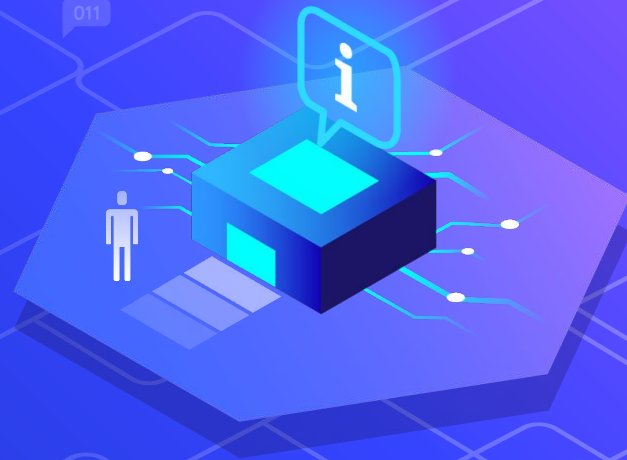
Snapshot : [45, max_int, max_int, 1, max_int]

New Sum




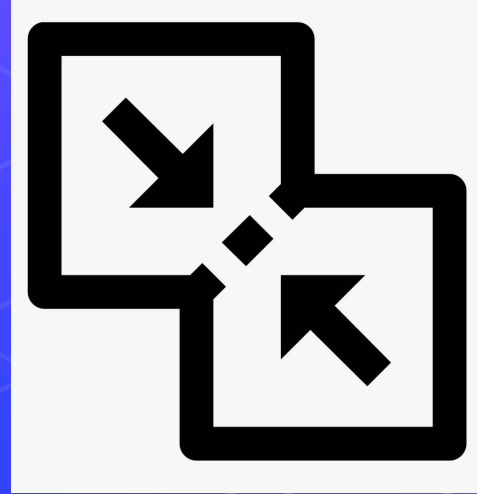
Data Reorganization

Lazy, Contention Free Merge



Merge: Criteria

- BP must be read-only 
 - Keep track of qualified BPs in bufferpool
 - Qualified BP is if BP is full
- Merge after every 500 updates, check qualified BP array



Merge: Process

- Open new thread using Python's Threading library and run in background
- Pull base page and associated tail pages into memory
 - Create a copy of the base pages
- Consolidate updates onto copied base pages
- After merge we change the key_dict to point to the new consolidated base pages



Indexing

Create Index & Drop Index



create_index()

- Iterate through base pages in disk
- Iterate through records in base page
 - Value: (base_path, record num)
- Add that value to a index dictionary:
 - (e.g value:[(path_base_i, record_num_j)])

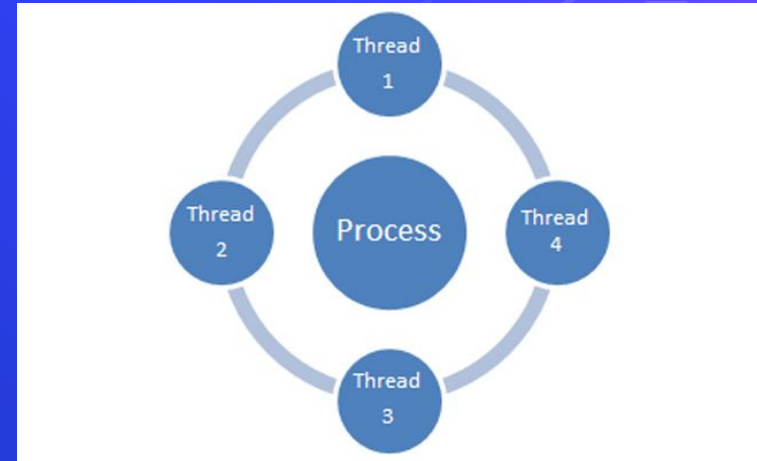
drop_index()

- ⬡ Deletes the dictionary for a given index



Future Goals & Plans

- Implement Multiple Threads for concurrent use of the L-Store



Extra resources





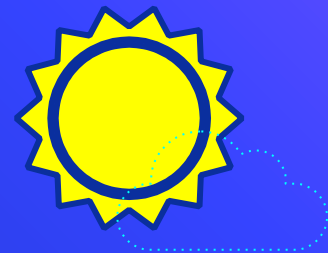
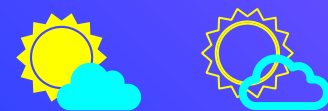
SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change fill color and opacity.
- Change line color, width and style.

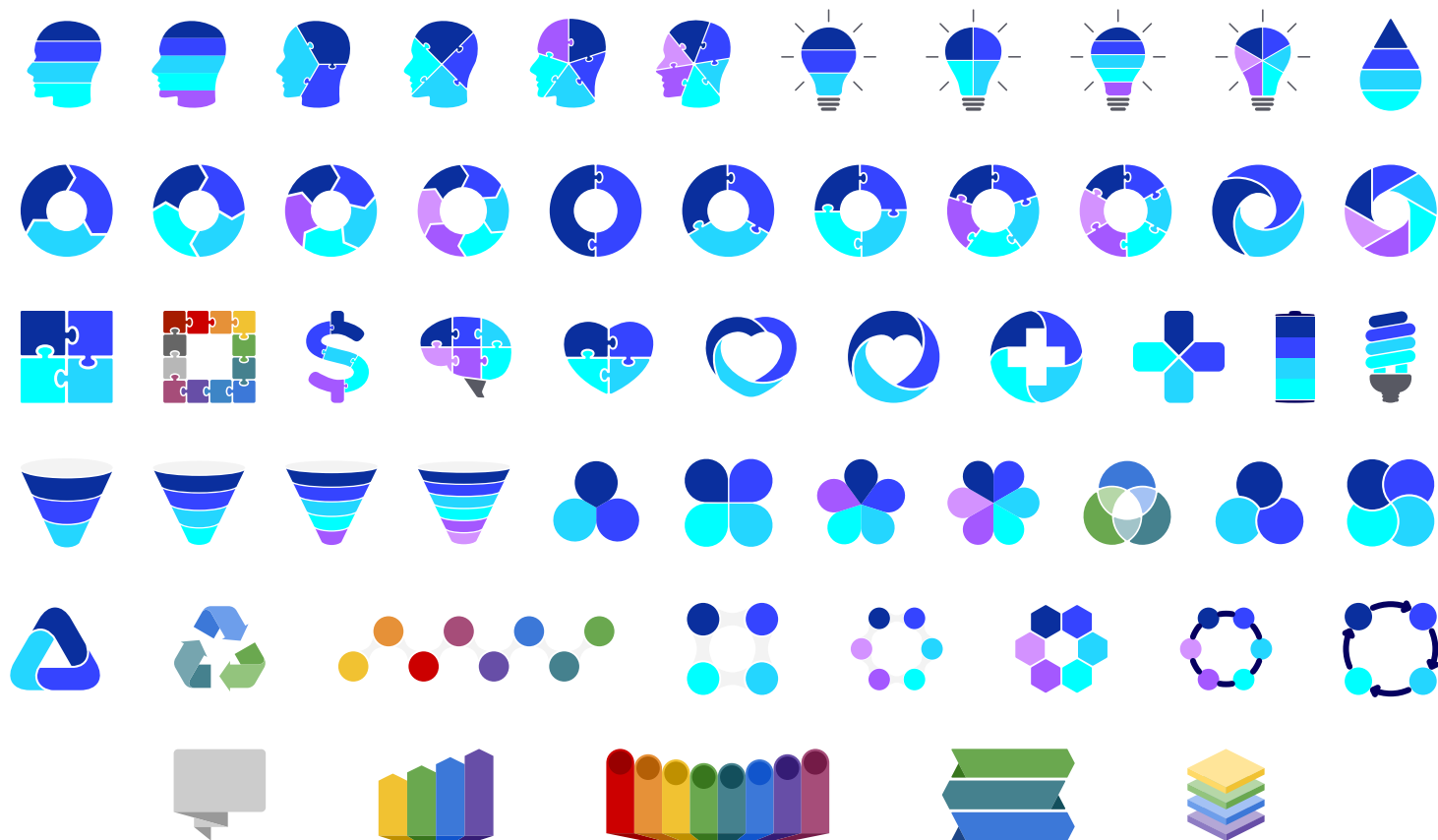
Isn't that nice? :)

Examples:



Find more icons at
slidescarnival.com/extra-free-resources-icons-and-maps

Diagrams and infographics



You can also use any emoji as an icon!
And of course it resizes without losing quality.

How? Follow Google instructions

<https://twitter.com/googledocs/status/730087240156643328>



and many more...