ECS165A Milestone 2

Winter 2022

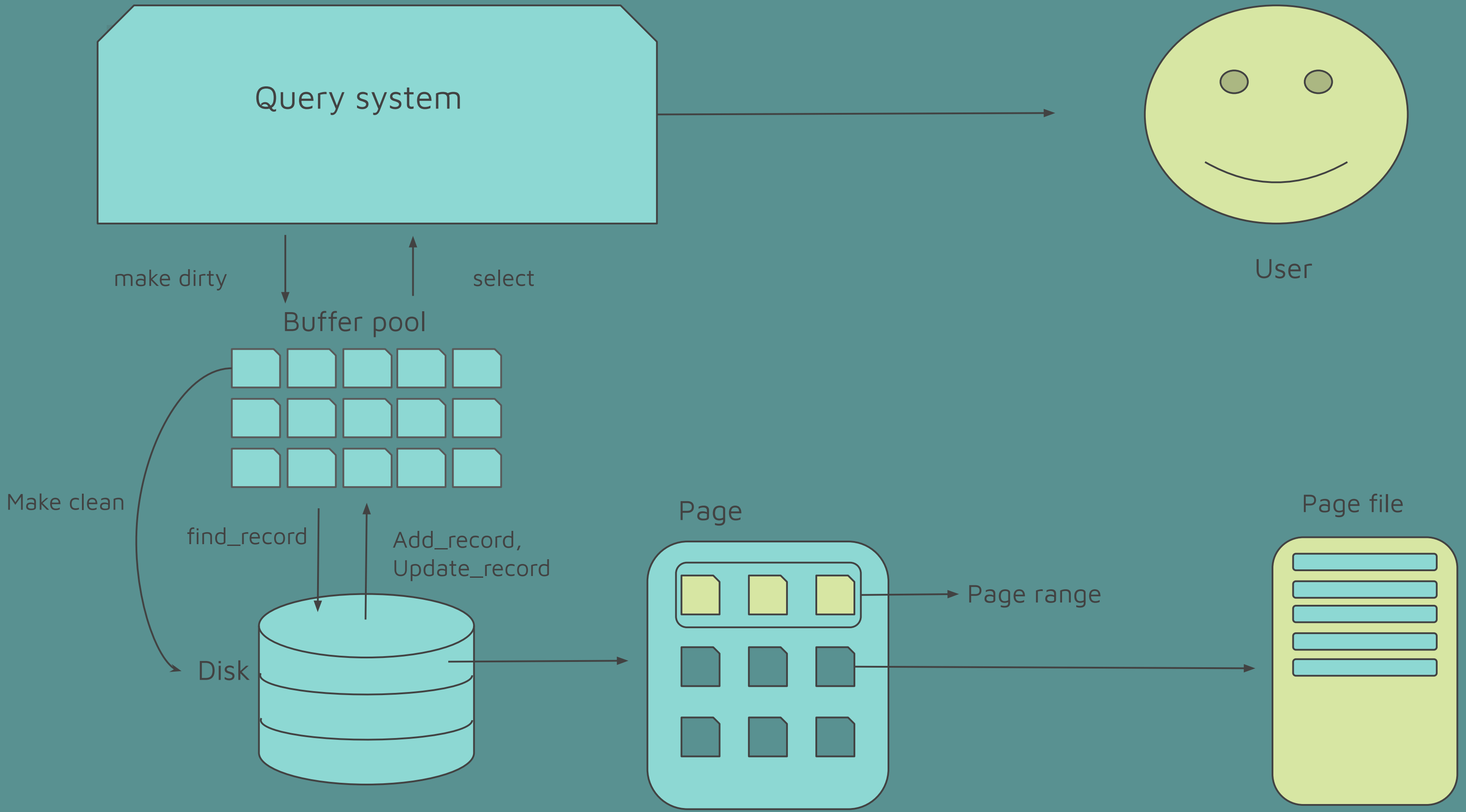
Michael Pitts, Jericho Mata, Ethan Yu, Ryan Cen and Yuyi Li

Goals & Objectives

There are three objectives we hope to achieve in this project:

- The first concern is balancing data in memory and disk.
- The second objective is to expand select index capabilities.
- The last objective is data reorganization through a contention-free merge.

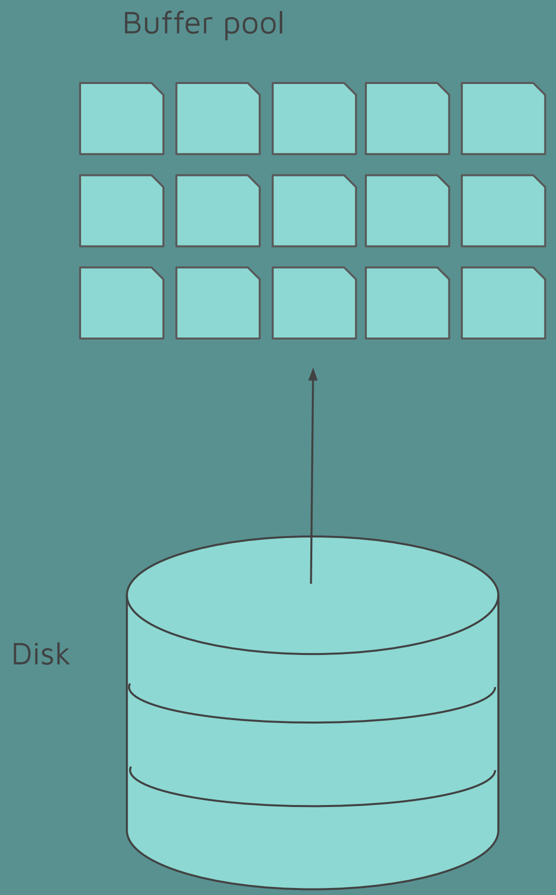
Updated Architecture



Bufferpool and Extension

- Due to limited memory space, we would like to save the more frequently used data(most recently used pages) in memory while keeping the less frequent ones in the disk.

- Pinning pages happens when we are currently using it.

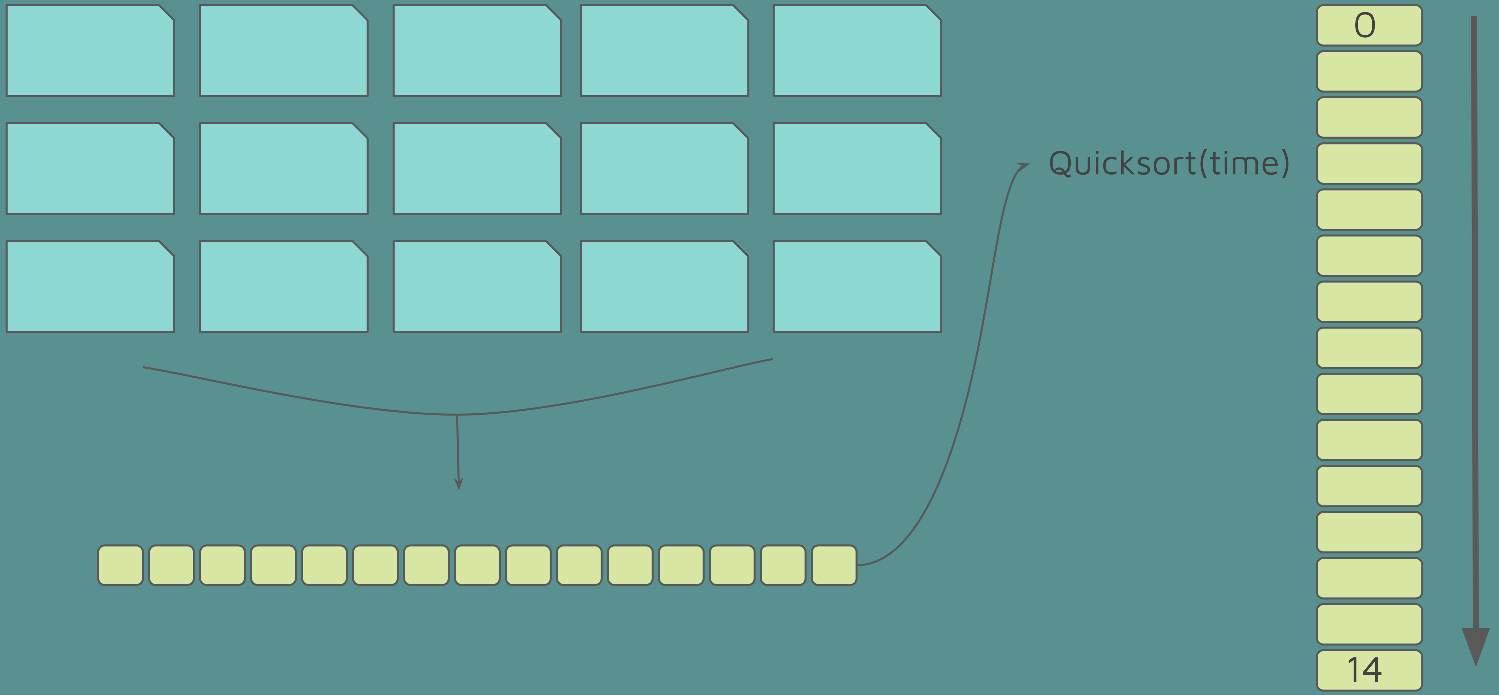


Bufferpool Eviction policy

- When it comes to evicting pages, we get rid of the least recently used ones. In the function evict_page(), we first clean the page first with make_clean(), then iterate through the bufferpool to find the least used, unpinned pages to evict.

- Lock up the disk whenever committing to it so we utilized the lock function alike to multithreading in the OS.

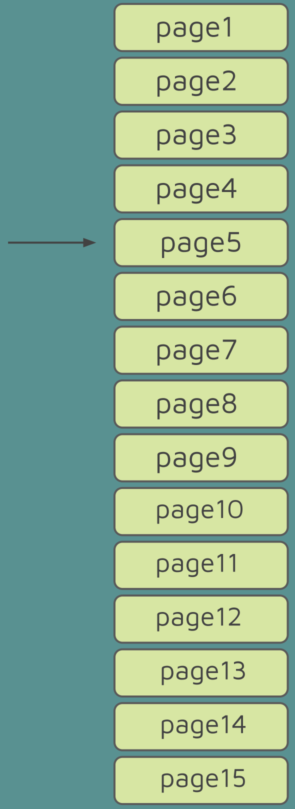
Eviction process



Eviction process

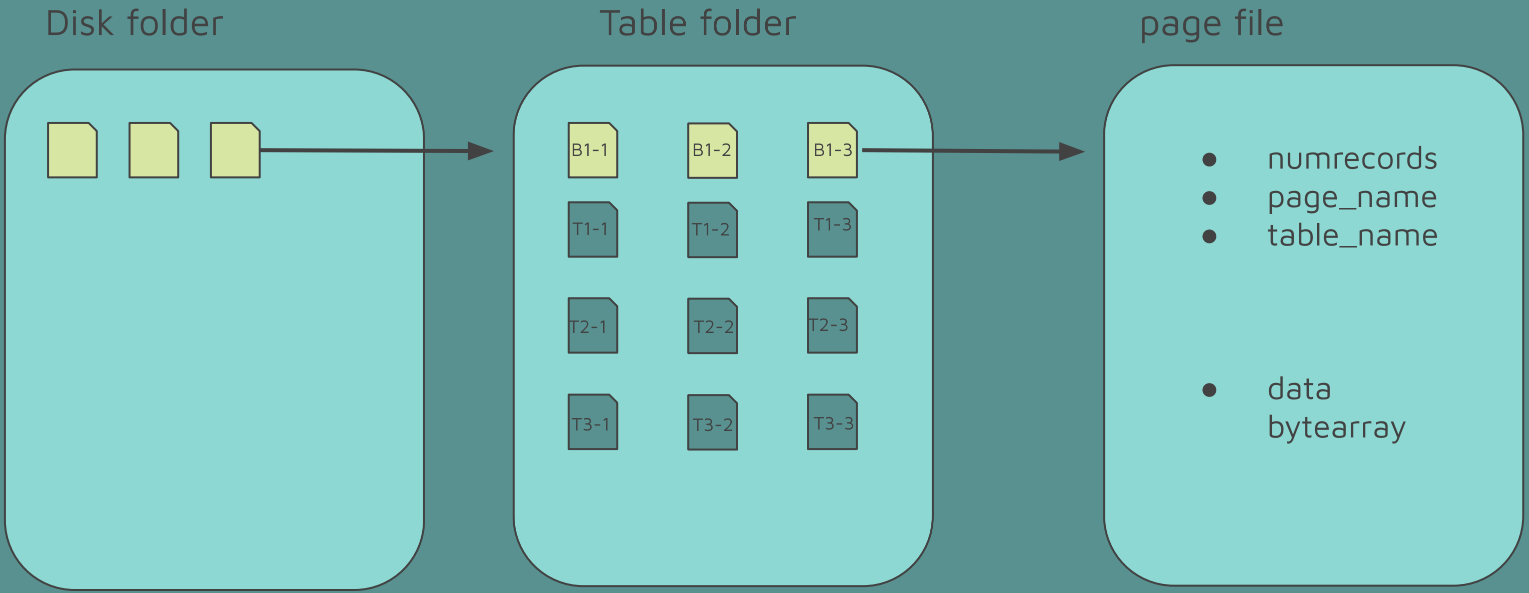
What's happening

Once the sorted array is extracted the function the runs down the list of pages from least recently used to most looking for an unpinned page. Once it is found that page is evicted.



```
For page in sorted_pages:
          if(page not in pinned_pages):
                    bpool.remove(page)
```

The Pseudo-disk:



Disk folder

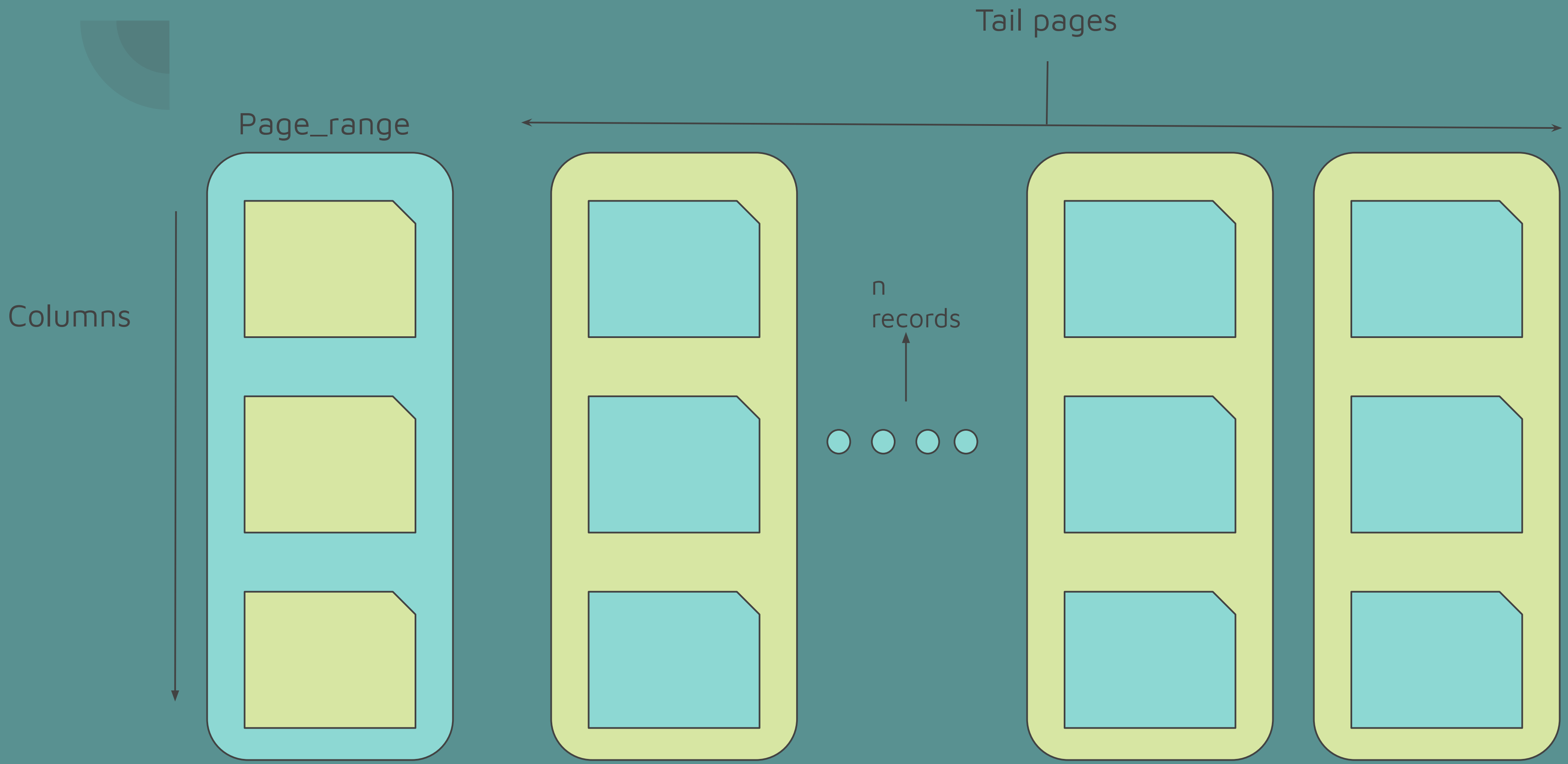
Table folder

B1-1 B1-2 B1-3
T1-1 T1-2 T1-3
T2-1 T2-2 T2-3
T3-1 T3-2 T3-3

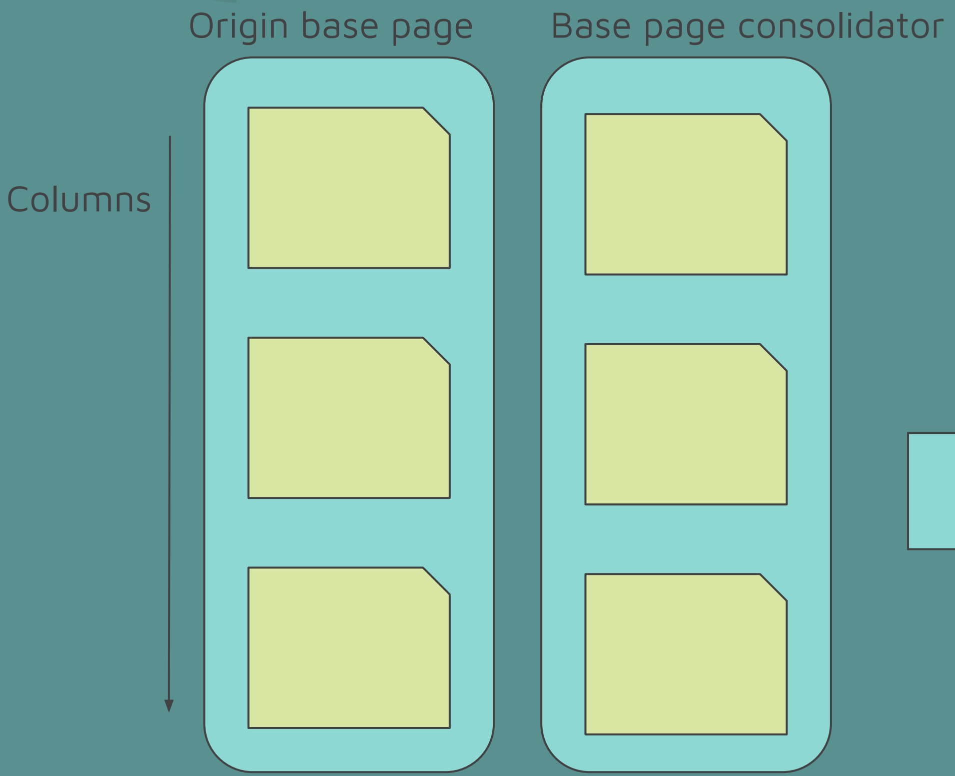
page file

- numrecords
- page_name
- table_name
- data bytearray

Merge function



Merge function Cont’l



Origin base page

Base page consolidator

Columns

The 999 tail pages are the compressed for each column in the page range.

Result of Merge:

Because of this merge the update and select function no longer have to wade through a sea of tail pages to find their target. The result of this is greatly improved performance.

Indexing

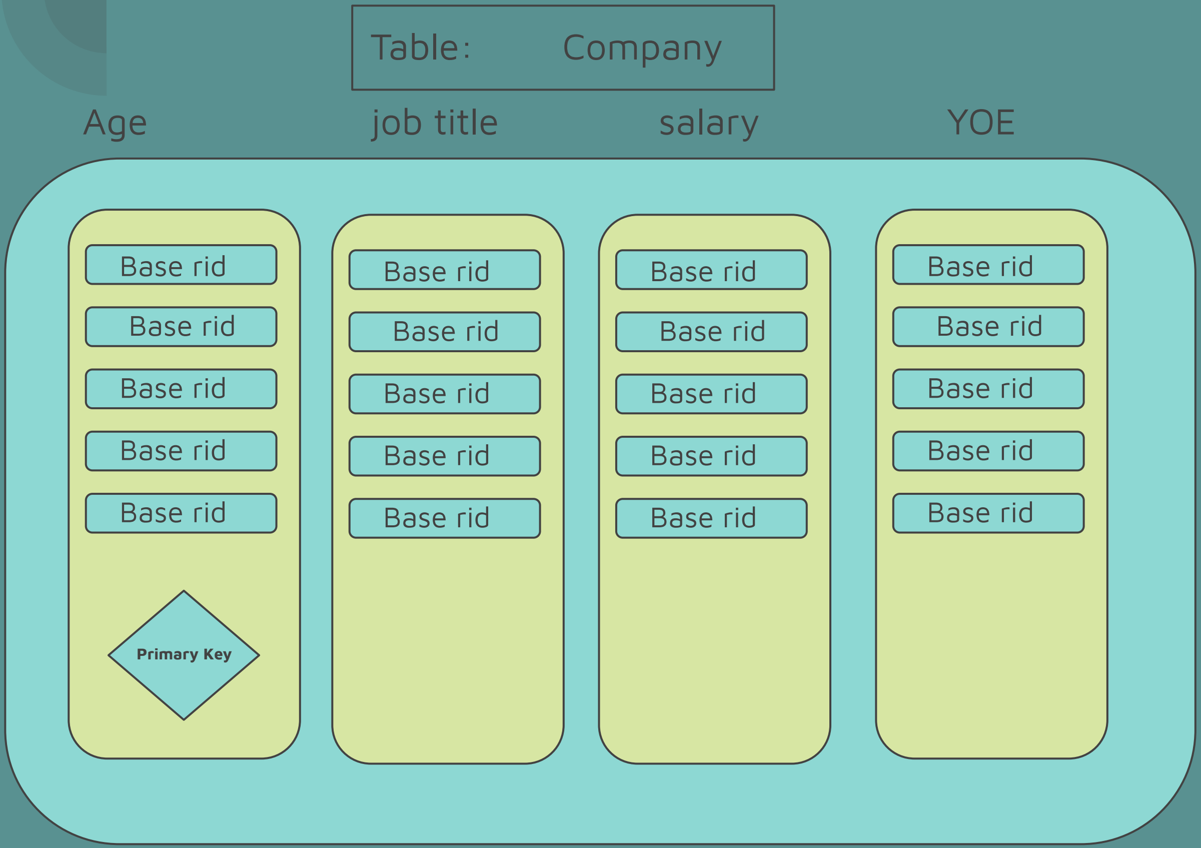


Table: Company

Age

Base rid
Base rid
Base rid
Base rid
Base rid

Primary Key

job title

Base rid
Base rid
Base rid
Base rid
Base rid

salary

Base rid
Base rid
Base rid
Base rid
Base rid

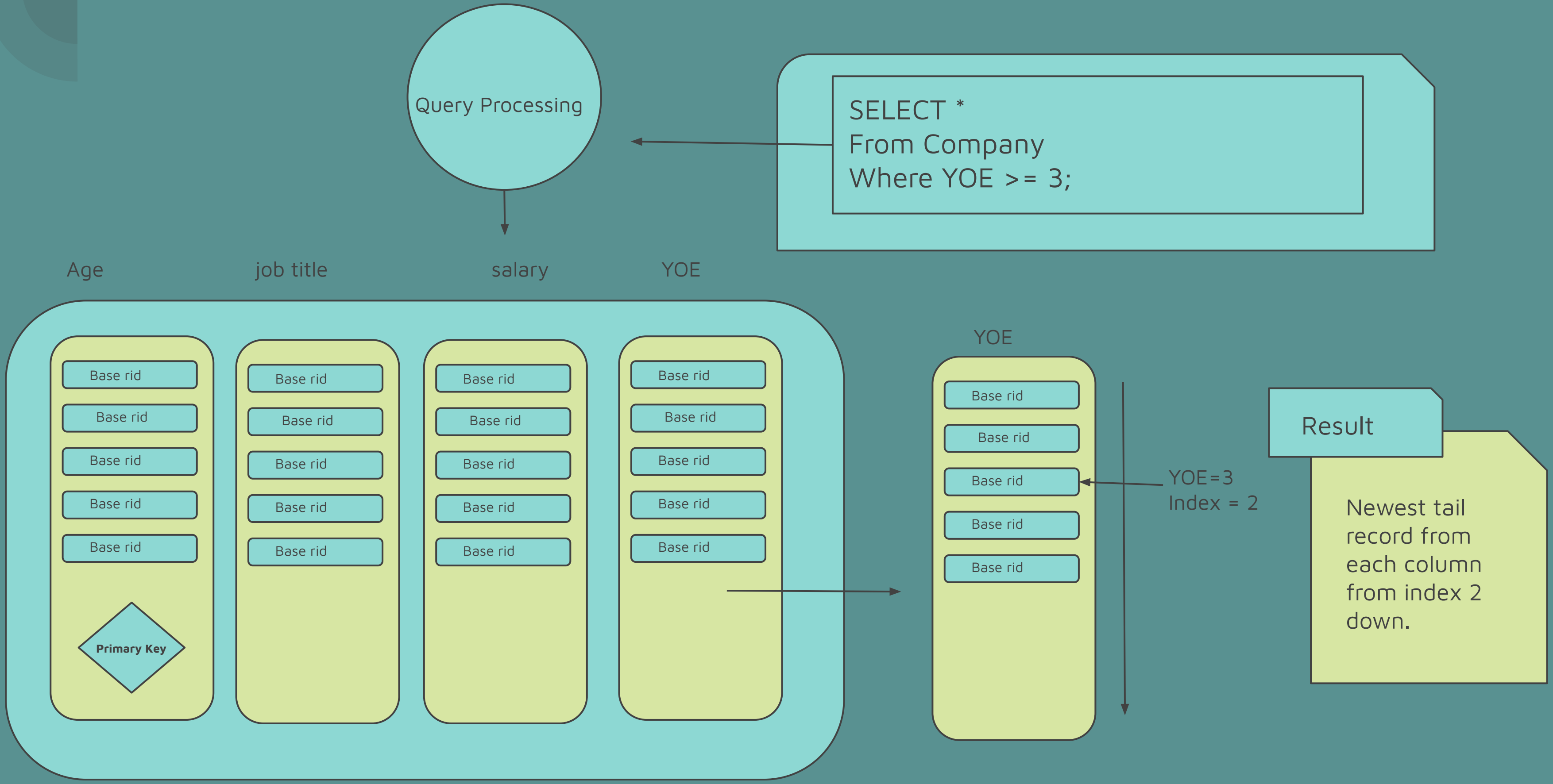
YOE

Base rid
Base rid
Base rid
Base rid
Base rid

What's happening

All the columns are filled with base rids that are sorted by their get_newest_value which returns the most updated tail page value.

Indexing Cont’l



```
SELECT *
From Company
Where YOE >= 3;
```

Newest tail record from each column from index 2 down.

Things to improve

- Speed, this can be improved through optimizing what data-structure we use throughout our system.

- Switch from a cumulative database to a non-cumulative database

Q & A Time