
# CMPS401 — Advanced Database Systems

## Assignment

11 Oct 2025 • $B^+$ Trees Insertion and Deletion

---

**Muhammad Sayed**

**We Juggle Things!**

# 1 Introduction

This assignment tasks you with implementing a file storage and indexing system using a B+ tree. The primary goals are to handle record storage in disk blocks, build a dynamic index, and observe the behavior of the B+ tree during insertion and deletion operations.

The required steps are as follows:

- Implement a framework for storing fixed-length records in blocks to simulate a basic file system.
- Read records from the provided CSV file into memory.
- Build a $B^+$ tree to index the records based on the SSN field.
- Perform a series of insertion and deletion operations on the $B^+$ tree.
- After each set of operations, visualize or print the state of the $B^+$ tree to demonstrate its structural changes.

---

# 2 Tree Operations

The following content describes the operations required for insertion and deletion in the $B^+$ tree.

## Insertion

The insertion operation in a $B^+$ tree proceeds as follows:

- Split the node if it exceeds the maximum capacity of pointers.
- **Split leaf node:**
  - Split the values such that the two new nodes have the same number of values or the right node has more values.
  - The smallest value in the right node goes to the parent (and stays also in the leaf).
- **Split internal node:**
  - Split the values such that the two new nodes have the same number of values or the right node has more values.
  - The middle value goes to the parent.
  - Middle value = Floor((#nodes + 1) / 2).

### Deletion

The deletion operation follows these steps, adapted from the presentation here:

- Find the key to delete.
- Go to the leaf node and remove it.
- If the leaf has too few keys (less than half full):
  - Borrow a key from a nearby leaf (Try left first).
  - If you can't borrow → Merge with that leaf.
- Update the parent's keys.
  - Update the separator if it changes.
  - If it contains the deleted key, replace it with the next key (smallest key in the right subtree).
    - If there is **no right subtree**, remove the key from the internal node.
- If the updated parent has is less than half full of pointers → fix it the same way (Borrow or Merge).
- If the root has only one child → make that child the new root (Tree level decreased by 1).
  - Note that's completely okay as root (and all internal nodes) are just there to provide directions.

---

## 3 Assignment Task & Requirements

### Problem Setup

Consider a disk with block size $\mathbf{B = 512\ bytes}$. A block pointer is $\mathbf{P = 6\ bytes}$ long. A file has $\mathbf{r = 10}$ **EMPLOYEE records** of fixed-length. Each record has the following fields: NAME (30 bytes), SSN (9 bytes), DEPARTMENTCODE (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes), BIRTHDATE (8 bytes), SEX (1 byte), JOBCODE (4 bytes), SALARY (4 bytes). An additional byte is used as a deletion marker. The B+ tree order for internal nodes is $\mathbf{p = 3}$ and for leaf nodes is $\mathbf{p_{leaf} = 2}$.

### Implementation Workflow

You are required to implement the following sequence of operations:

1. Load all records from the provided CSV file into memory.
2. Insert the first **10** records into the file blocks.
3. Show the current state of the blocks saving the records.
4. Create a B+ tree index on the **SSN** field for the initial 10 records.
5. Print the current status of the B+ tree.
6. Insert the records with the following original line numbers from the CSV file: **27, 14, 22**.
7. Show the updated B+ tree index.
8. Delete the records with the following original line numbers from the CSV file: **11, 6, 3**.
9. Show the final B+ tree index after deletions.
10. Show the final state of the file blocks.

---

## 4 Evaluation & Submission

### Grading Criteria

Your submission will be evaluated based on the following criteria:

- **Data Structures and Classes:** Correct implementation of classes for records, blocks, and the B+ tree.
- **Insertion Algorithm:** Correct implementation of the B+ tree insertion logic, including node splits.
- **Deletion Algorithm:** Correct implementation of the B+ tree deletion logic, including borrowing and merging.
- **Printing and Visualization:** The function for printing the tree and blocks should be user-friendly and clearly represent the state of the data structures.
- **Bonus Points:** Additional effort, such as advanced visualization, error handling, or performance optimizations, will be rewarded with bonus points.

## Marking and Teamwork

- The assignment is graded out of **10 Marks**.
- You should work in teams of **4**, the same teams as your course project.
- Deadline: **Saturday, 1 November 2025, 11:59 PM**.
- Submit your information, code and visualizations on google classroom.

Contact Information

For any questions, please feel free to reach out via email:
cmpsy27@gmail.com
muhammad.sayed@eng.cu.edu.eg