Assignment: Database Management System

Amit Kumar Dhar

November 17, 2025

Assignment Task Overview

You are required to implement a B+ tree index as in databases. The index will be stored in a file on disk. Your program would be responsible for writing, reading as well as searching. For ease of implementation, you may assume that the key on which B+ tree is indexed is of integer type. Similarly, for ease of implementation, you may assume that the data of each tuple is store in a fixed size byte array of 100 bytes. The page size for the implementation should be 4096 bytes. Since there are no buffer manager, you may use memory-mapped I/O (or if you want to experiment, you can implement your own buffer manager).

Your program can start either with an empty file (the first run) or with an existing index in file. In case some data exists in the file, your program should read and write the data accordingly. Note that the leaf nodes are expected to store both keys and the complete corresponding tuple. Also note that there is NO guarantee that the index could be store wholly in the RAM (in fact this is going to be the case for the test input). All changes thus needs to be written to file and dynamically read when needed.

Assignment Tasks

Write a program in your favourite language which provides the following API.

1. Writing API

   (a) A function named `writeData(key,data)` which takes a key and corresponding tuple as per previous explanation and inserts them in the index. It should return true/1 if the insertion is successful and saved. Otherwise it returns false/0.

   (b) A function named `deleteData(key)` which takes a key and deletes it from the index. It should return true/1 if the deletion is successful and saved. Otherwise it returns false/0.

2. Reading API

   (a) A function named `readData(key)` which takes a key as per previous explanation and searches them in the index. It returns the tuple bytes corresponding to the given key if it exists. If the key does not exists in the index then it returns NULL/0.

   (b) A function named `readRangeData(lowerKey,upperKey,n)` which takes two keys as per previous explanation and searches for all keys between them (inclusive of the given keys) in the index. It returns the array of tuple bytes corresponding to the given keys (n represents the length of the array). If there are no key in the range exists in the index then it returns NULL/0.

Testing with driver

Write a driver program (a separate program) which can call these APIs and verify that your program is working correctly.

Submission Instructions

- Submit the program files in a zipped file along with the following:

  1. source file(s)
  2. `requirements.txt` (if applicable)
  3. `README.md` explaining setup, compilation and execution steps. Extra marks for creating a make file and writing linux style documentation for the APIs

  Be careful, if the stated instructions do not result in successful compilation and execution on a standard ubuntu desktop, no marks will be given.

Submission Instructions

- Submit the program files in a zipped file along with the following:

  1. source file(s)
  2. `requirements.txt` (if applicable)
  3. `README.md` explaining setup, compilation and execution steps. Extra marks for creating a make file and writing linux style documentation for the APIs

Marking Scheme

- Successful compilation and execution of program with test input and with the help of your instructions will get you 50% marks.
- The rest marks will be divided equally in the interval between the running time of the fastest and the slowest submission. You will get marks based on how close is the running time of your program to the fastest submission.
- Any attempts to adopt unfair means will result in zero marks.