Homework: B+tree Index (w5)

New Attempt

Due May 10 by 11:59pm **Points** 9 **Submitting** a file upload

Introduction

The objective of this assignment is to learn how to implement index structures on data stored in the external memories.

What you must do

Assume that we have a relation **Employee(id, name, bio, manager-id)**. The values of **id** and **manager-id** are integers each with the fixed size of 8 bytes. The values of **name** and **bio** are character strings and take at most 200 and 500 bytes, respectively. Note that as opposed to the values of id and manager-id, the sizes of the values of **name** and **bio** are not fixed and are between 1 to 200 (500) bytes. The size of each page (block) is 4096 bytes (4KB). The size of each record is less than the size of a page. Using the provided skeleton code with this assignment, write a C++ program that creates a dense and unclustered B+tree index file for relation **Employee** using attribute **id**. Your program must also enable users to search the created index by providing the id of a record.

• Your program must first read an input **Employee** relation and build a B+tree index for the relation using attribute **id**. The input relation is stored in a CSV file, i.e., each tuple is in a separate line and fields of each record are separated by commas. Your program must assume that the input CSV file is in the current working directory, i.e., the one from which your program is running, and its name is **Employee.csv**. We have included an input CSV file with this assignment as a sample test case for your program. Your program must create and search B+tree indexes correctly for other CSV files with the same fields as the sample file.

• You may create a B+tree index by reading the input file and repeatedly inserting the input records in the data file and its relevant information in the index file. To do this, you may use the fragments of the the program you have developed in the previous assignment to read input records and store them in the data file. The data file is a heap (unsorted) file of variable-length records and pages on the external storage. They are also explained in Sections 9.7.2 and 9.6.2 of Cow Book, respectively.

• Your program must insert the information about **ids** of the input records in the B+tree index file. The program mustdo this both while creating the data file. It must store the B+tree index in a file with the name **EmployeeIndex** on the current working directory. Each node of the B+tree is a record in this file. The structure of these records must follow the structure of B+tree nodes explained in the lecture. You may treat these records as both fixed-length records and use the data structure described in the class to store fixed-length records and their corresponding pages to store them in the index file. The pointer in the root and internal nodes keep addresses of referred nodes in the B+tree index file. The pointers in the leaf nodes of the index must keep the addresses of their corresponding records on the created data file.

• During the index creation, your program may keep up to three pages plus the root of the index in main memory at any time. The submitted solutions that use more main memory will not get any points.

• You must set the degree (d) of the index to 4.

• After finishing the index creation, your program should accept an **Employee id** in its command line and search the index file for all records of the given id.

Like index creation, your program may use up to three pages plus the root of the index in main memory at any time. The submitted solutions that use more main memory

will not get any points for implementing lookup operation. The user of your program may search for records of multiple ids, one id at a time.

• Each student has an account on **hadoop-master.engr.oregonstate.edu** server, which is a Linux machine. Your should ensure that your program can be compiled and run on this machine.

You can use the following bash command to connect to it:

```
> ssh your_onid_username@hadoop-master.engr.oregonstate.edu
```

Then it asks for your ONID password and probably one another question. To access this server, you must be on campus or connect to the Oregon State VPN.

• You can use following commands to compile and run C++ code:

```
> g++ -std=c++11 main.cpp -o main.out
  > main.out
```

Necessary files

Input file:**Employee-2.csv (https://canvas.oregonstate.edu/courses/1939345/files/99023162?wrap=1)** ⬇
(https://canvas.oregonstate.edu/courses/1939345/files/99023162/download?download_frd=1)

Following files contain the skeleton code to generate the required EmployeeIndex & DataFile. You may make changes to these files adhering to the assignment requirements.

main.cpp (https://canvas.oregonstate.edu/courses/1939345/files/99069298?wrap=1) ⬇
(https://canvas.oregonstate.edu/courses/1939345/files/99069298/download?download_frd=1)

classes.h (https://canvas.oregonstate.edu/courses/1939345/files/99069300?wrap=1) ⬇
(https://canvas.oregonstate.edu/courses/1939345/files/99069300/download?download_frd=1)

What to turn in

The assignment is to be turned in before Midnight (by 11:59pm) on May 10. You may turn in the source code of your program through Canvas.

The assignment may be done in groups of two students. Each group may submit only one file that contains the full name, OSU email, and ONID of every member of the group.

Grading criteria

The programs that implement the correct algorithm, return correct answers, and do not use more than allowed buffers will get the perfect score.

The ones that use more buffer than allowed will not get any points. The ones that implement the right algorithm but return partially correct answers will get partial scores.

B+ Tree Index

Criteria	Ratings					Pts
Description of criterion	**9 pts** **Full Marks**	**7 pts** **Minor Error** Lookup did not return all the IDs we were searching for.	**4.5 pts** **Partial Credit** EmployeeIndex created properly but Lookup implementation did not match assignment requirement.	**3 pts** **Major Error** Your program do not return any queries correctly. Your EmployeeIndex file structure shows that you are reading the EmployeeIndex file sequentially for Lookup. It defeats the purpose of having a B+ tree structure.	**0 pts** **No Marks** You did not meet the assignment requirements	9 pts
						Total Points: 9