CS 7280: Special Topics in Database Management Spring 2024

Project 1: Indexing Structure for NoSQL database

Objectives:

1. Understand Indexing Structure
2. Work with NoSQL database
3. Become familiar with database algorithm
4. Gain experience with algorithms in DBMS and NoSQL

Due: Wednesday, February 14, 2022. 11:59:59 PM (PST)

This will be a group project by 2 members. In this project, you are going to implement one of indexing structures in any programing language.

- B-Tree
- HashMaps
- T-Tree (or T*-Tree): Main memory database
- O2-Tree: A Shared Memory Resident Index in Multicore Architectures
- Bloom filter index: fast search algorithm for multi-hash function mapping
- LSM tree indexing: Log Structured Merge (LSM) tree indexing
- Others

Your task is to implement a main memory B-tree indexing structure including the following functions (methods):

- *Lookup(int keyalue)*: find the specified value. If the value exists, returning value is *True*.
- *Insert(int keyvalue)*: insert the specified value.
- *Display(int node)*: print out the indexing tree structure under specified node.

The value storage is limited to simple integers. Instead of using of objects to simulate block operations, the framework utilizes a simple array referencing the blocks, which is dynamically growing if the number of occupied blocks increases.

Suggested Project Steps

We suggest that you take the following steps:

B-Tree

1. Select your indexing structure. You can choose one from suggested list or find another one by considering:
   - Your index will be used for index of NoSQL database in Phase 2.
   - The data structure for NoSQL database will be basically Key-Value, and Key will be *keyvalue* in your index.

- In Project 1, your index will be main-memory index, but in Phase 2, it should be an index file consisting of index blocks.

2. Research on your selected indexing structure algorithm including lecture note and given papers. However, you are encouraged to research on the algorithm by yourself. You do not want to start this project without understanding what the node should have, and how the index is kept balanced.
3. You need to design 'Node' data structures, i.e., Block eventually. We are going to use the same data structure for any types of nodes. The *keyvalue* stands for key as well as value in general indexing structure.
4. Write and test your code for searching values from your index, i.e., *Lookup( )*.
5. Write and test your code for inserting values, i.e. *Insert( )*.
6. You can create a test program to test the index with various test data.
7. Blocking factor (bfr or node size) can be fixed in this phase. However, in Phase it can be vary depending on your data. You can set default value as 5.

Some Guidelines

- Implement your index without using any program language specific data structure. You can use integer arrays in Java or list in Python.
- DO NOT copy and paste any existing codes from internet. If you are caught, zero credit for the entire projects.
- Take a look at lecture notes or papers for more information about indexing structure.
- Create Github repository and upload all your codes.

Write-up

You should create a Github repository including write-up and your code. Your write-up should include any known bugs, limitations, and assumptions in your program. This write-up should be in text-format and titled as 'README.md' in your github repository. TA will use the 'README.md' file to compile (or install) and run your program. If TA has trouble with your program then she/he will contact you to makeup it.

Submission

You will submit Github link for your project repository to Canvas. Your submission should include at least (1) your sources, and (2) README.md.

Make sure your name and NEU ID are listed in both your README.md and source codes. You may resubmit your program at any time. The submission time of the latest submission will be used for determining whether the assignment is on time or not. Late submissions will be accepted at a penalty of 15 % per day (up to 2 days). In other words, it may pay you to do this project early on the off chance that something prohibits your submitting it in a timely way. If your program is not working by the deadline, submit it anyway and review it together with TA for partial credit. Do not take a zero on any

project just because the program isn't working yet. If you have trouble getting started, ask the professor or the TA.

Demonstration

After the project is submitted, each student should demonstrate it with TA. The sign-up sheet will be provided before due date, and the student can choose 10 minutes time slot for the demonstration.

When: TBA

How: Online

Grading (100%)

Correctness of indexing (insert and search)	60%
README file	15%
Source code (comments especially)	15%
Whether you follow the instruction or not	10%

To receive full credit for comments in the code you should have headers at the start of every module, subroutine, or function explaining the inputs, outputs and function of the module. You should have a comment on every data item explaining what it is about. (Almost) every line of code should have a comment explaining what is going on. A comment such as /* add 1 to counter */ will not be sufficient. The comment should explain what is being counted.

heap file:

add at the end…

extras:
potentially implement with IP, mySQL, client side as well