CSCI 4370: Database Management

Project 3: Index Structures - Linear Hashing

Use indices [TreeMap (in Java), HashMap (in Java) and LinHashMap (your own implementation)] to speed up the processing of Select and Join in your earlier Project 1 implementation. Indices must be integrated and used in the Table class.

Step 1: Implement LinHashMap methods:

Search for "// T O B E I M P L E M E N T E D" tags and only implement them. It has already been partially implemented in LinHashMap.java (at eLC) and what you need to do is to fully implement this index structure. The main method of LinHashMap.java is used for testing; however, since functions are partially implemented, it will not give the proper output. So, you should be able to see the desired output after the completion of implementation. Download the starter source code from the eLC and compile it. Upon making sure that it is working properly, you can start implementing. After implementation, you can compile and run your code to verify if the outputs are correct.

Step 2: Implement Table.java methods: This file contains Table class which implements relational database tables. You need to implement indexed-select and indexed-equi-join **only** for performance evaluation. For indexed-select, it is simple (no more than 5 lines of code) and non-indexed version is provided for evaluation(noIndexSelect). Non-indexed version of equi-join is also provided to make the implementation quick and easy(noIndexjoin). Search for "// T O B E I M P L E M E N T E D" tags. Do not worry about other unimplemented methods.

Next Steps: Performance Evaluation:

Goal: Compare the speed of (a) NoIndex, (b) TreeMap, (c) HashMap and (d) LinHashMap. The code for (a) NoIndex has already been provided in the Table.java file.

A workflow for this step may look like:

- Study the TestTupleGenerator file. It defines a "Student Registration Database" and prints tuples for each entity.

- Modify the code such that instead of printing the tuples on the terminal, they are stored as rows of Table objects.

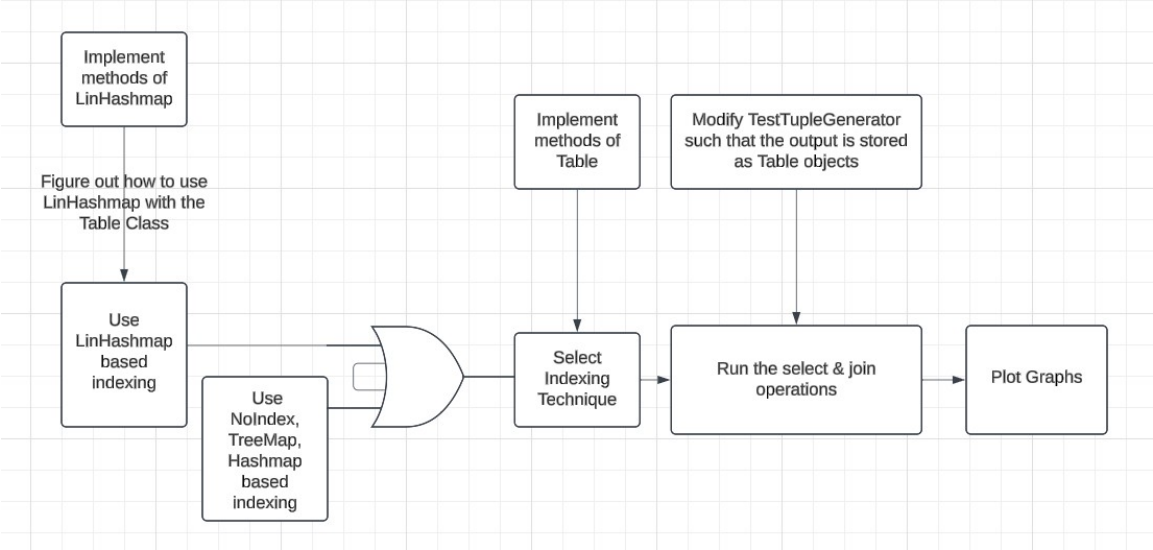
- Run the following two operations on the tables for each of the indexing mechanism mentioned above (a through d).

    (i) Select Operations: "σ [id = v] (Student)"

    (ii) Join Operations: "Student join [id = studId] Transcript".

Plot the speed/performance in terms of response time in ms[on the y-axis] vs. number of tuples in total in the table(s)[on the x-axis]. Try the following total number of tuples: 500,

1000, 2000, 5000, 10000 [These will be the points on the x-axis]. You will have 8 graphs in total as you have 2 operations (join & select) and 4 indexing techniques (a through d).



What to submit: Please submit

- all source code
- a readme file
- The data points as a single excel sheet
- The 8 graphs
- other docs indicated by the project submission guidelines

The readme file should contain: your names, how to compile and run your code and other specifications you want to make (including your database access information for testing purposes). Please pack all your files in a zip package with the file name: project3_ last names of your group members. For example: project3_Miller_Arpinar.zip

How to Submit:

Submit your ".zip" file using ELC. Only team leaders need to make a submission.

Do not place your solution on a public web site. Submit your own work and follow the course misconduct policy.