Project 1

Due: 5PM 05 March 2023 (Sunday)

1. **Introduction.** This project is to design and implement the following two components of a database management system, storage and indexing.
   (1) For the storage component, the following settings are assumed.
   - a fraction of main memory is allocated to be used as disk storage for simplicity (and thus there is no need to worry about the buffer management); In the following, by disk storage, it refers to this allocated memory storage.
   - the disk capacity could be 100 - 500 MB[1] (depending on your machine's main memory configuration);
   - the disk storage is organized and accessed with a block as a unit; Note that since this disk storage is the allocated memory storage and can in fact be accessed with a byte/word as a unit, some mechanism needs to be designed here to simulate accesses with a block as a unit.
   - the block size is set to be 200 B;

   (2) For the indexing component, the following settings are assumed.
   - a B+ tree is used;
   - for simplicity, the B+ tree is stored in main memory, with each node is stored in a memory region bounded by the block size (this simulates the case that each B+ tree node could be stored in a block on actual disk when necessary);

2. **Implementation and Experiments.**
   (1) (10 marks) Design and implement the storage based on the settings described in Part 1. C/C++ is recommended for this project, but other programming languages including Java and C# are also acceptable.
   - Describe the content of a record, a block, and a database file;
   - Specify your choices for various options (e.g., spanned or non-spanned; sequencing vs non-sequencing, etc.) and provide justifications;
   - This part will be assessed based on the design description, codes, and the experiments based on the implementation (for verification).

   (2) (25 marks) Design and implement indexing components based on the settings described in Part 1.
   - Your implementation should be able to support data with duplicates of keys; Please present your design clearly and explain how it is able to support duplicates of keys;

[1] 1MB = 2^20 Bytes

- This part will be assessed based on the design description, codes, and the experiments based on the implementation (for verification).

(3) (5 marks) Experiment 1: store the data (which is about IMDb movives and described in Part 4) on the disk (as specified in Part 1) and report the following statistics:

- the number of records;
- the size of a record;
- the number of records stored in a block;
- the number of blocks for storing the data;

(3) (10 marks) Experiment 2: build a B+ tree on the attribute "numVotes" by inserting the records sequentially and report the following statistics:

- the parameter n of the B+ tree;
- the number of nodes of the B+ tree;
- the number of levels of the B+ tree;
- the content of the root node (only the keys);

(4) (10 marks) Experiment 3: retrieve those movies with the "numVotes" equal to 500 and report the following statistics:

- the number of index nodes the process accesses;
- the number of data blocks the process accesses;
- the average of "averageRating's" of the records that are returned;
- the running time of the retrieval process (please specify the method you use for measuring the running time of a piece of code)
- the number of data blocks that would be accessed by a brute-force linear scan method (i.e., it scans the data blocks one by one) and its running time (for comparison)

(5) (10 marks) Experiment 4: retrieve those movies with the attribute "numVotes" from 30,000 to 40,000, both inclusively and report the following statistics:

- the number of index nodes the process accesses;
- the number of data blocks the process accesses;
- the average of "averageRating's" of the records that are returned;
- the running time of the retrieval process;
- the number of data blocks that would be accessed by a brute-force linear scan method (i.e., it scans the data blocks one by one) and its running time (for comparison)

(6) (15 marks) Experiment 5: delete those movies with the attribute "numVotes" equal to 1,000, update the B+ tree accordingly, and report the following statistics:

- the number nodes of the updated B+ tree;
- the number of levels of the updated B+ tree;

- the content of the root node of the updated B+ tree(only the keys);
- the running time of the process;
- the number of data blocks that would be accessed by a brute-force linear scan method (i.e., it scans the data blocks one by one) and its running time (for comparison)

Note. The presentation of the report will be assessed and the weightage is 15 marks. **For the representation of the experimental results, please present them in the form of tables for easier checking. Please make the presentation of your report clean, sorted, and consistent as much as you can. These would be considered for assessing the presentation of the report.**

3. Materials to submit including:

A report including:

(1) Design of the storage component and the B+ tree component. It is suggested to use some figures to illustrate the designs.
(2) Results of the experiments in Part 2;
(3) The contribution of each group member (presented at the first page of the report); and
(4) **Source code** (You must attach an installation guide to ensure that your code can be run successfully. You will not receive any credit if your code fails to execute.)

4. Data. The data contains the IMDb rating and votes information for movies

- tconst (string) - alphanumeric unique identifier of the title
- averageRating – weighted average of all the individual user ratings
- numVotes - number of votes the title has received

The first line in each file contains headers that describe what is in each column. The data could be downloaded via this link:
https://www.dropbox.com/s/c04kfatnd9lrtx9/data.tsv?dl=0

5. Submission policy.

(1) All submissions should be uploaded to NTULearn (a submission slot shall be created later on).
(2) Late submissions will be penalized by 5% deduction per day for at most 7 days. Beyond 7 days after the deadline, no submissions will be accepted.
(3) It is not allowed to copy or refer to public code repositories. Strict plagiarism will be conducted. Any found plagiarism will mean a failing grade and be subject to further disciplinary actions. Some groups may be asked to demonstrate/explain their codes.