CS 2001-Data Structures (Fall 2022)
Semester Project

Group work (max 2 members allowed)

Submission Guidelines:

- **Make submission through google classroom only having a zipped folder namely DS-Fall2022- ClassProject-Roll#1_Roll#2.zip. This folder should contain all of your project files including source code, brief project report, instruction/help file etc.**
- **Virus infected or corrupted files / submissions will not be graded in any case.**
- **Properly follow these guidelines as they have marks.**

Deliverables:

- You should submit the Properly commented complete source code, readme file, brief report, and sample output/result of queries/questions.

Important Note:

- Plan your work on daily basis so that you do not miss deadline as this project need much effort and smart approach.
- Deadline is hard as simply **No credit for late submissions and <u>no submission means 'F'</u>.**
- Your code should be proper commented.
- Cheating will not be tolerated under any circumstances.
- Do not allow others to copy your work. You are totally responsible for managing the access rights on your code etc.
- Your work will be evaluated on quality and the extent to which you have utilized the knowledge that has been imparted during semester.
- **Individuals within a team/group may get uneven grades based on their contribution and level of understanding. So stay prepared for any type of evaluation methodology.**

Motivation:

In this project, you will mainly use B-Tree and Graph Data Structures' concepts to implement index/indices on different attributes/columns for efficient search; to answer queries, perform complex queries/joins, and perform social circle analysis on given dataset(s). Provided that you do this project at your own, you will practically understand the advantages and applications of the subject data structures while working with reasonably large amount of data and hence getting an opportunity to prepare yourself for a successful professional career in Computing.

Project Part 1

Background:

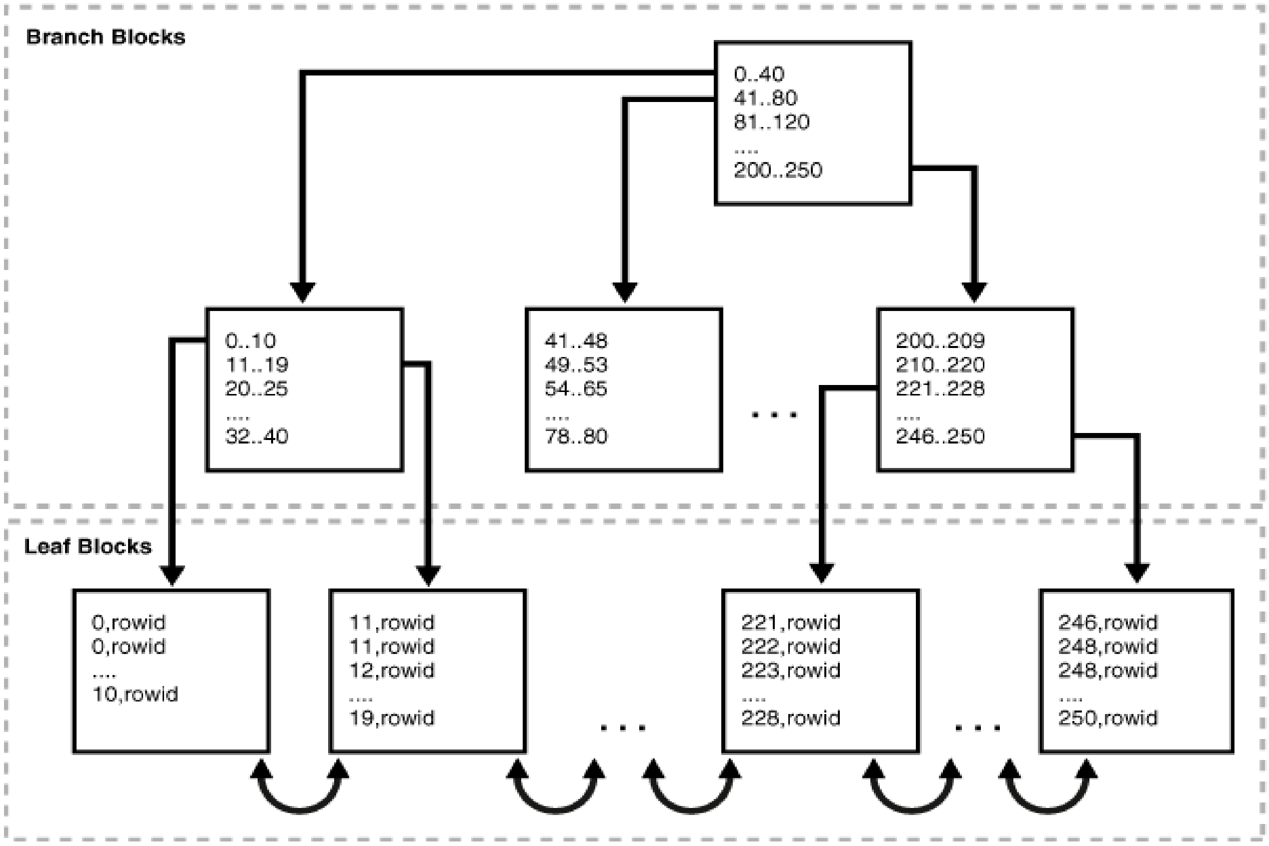
In computer science, a **B-tree** is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time. The B-tree is a generalization of a binary search tree in that a node can have more than two children. Unlike self- balancing binary search trees, the B-tree is optimized for systems that read and write large blocks of data from/to hard-disks. **B-trees are a good example of a data structure for external memory and it is commonly used in databases and file-systems.**

A **database** is an organized collection of data, that supports updating, retrieving, and managing the data. The data can be anything, from names, addresses, pictures, and numbers. Databases are used every day and everywhere, a University might maintain a database of employees, students and their grades (like Flex). To be useful and usable, databases must support operations, such as retrieval, deletion and insertion of data. Databases are usually large and cannot be maintained entirely in memory (RAM). **B-trees are often used to index the data and to provide fast access.** Searching an unindexed and unsorted database containing n key values will have a worst case running time of O(n). If the same data is indexed with a B-Tree, the same search operation will run in O(log n). To perform a search for a single key on a set of one million keys (1,000,000), a linear search will require at most 1,000,000 comparisons. If the same data is indexed with a B-tree of minimum degree 10, 114 comparisons will be required in the worst case.

An **index** is an optional structure, associated with a table that can mostly speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table very quickly. **Indexes** are one of many means of reducing disk I/O and hence improving the execution time of programs/software.

B-trees are the most common type of database index. A B-tree index is an ordered list of values divided into ranges. By associating a key with a row or range of rows, **B-trees provide excellent retrieval performance for a wide range of queries, including exact match and range searches. Oracle** database engine do provide implementation of B-Tree index. https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CNCPT1170

You can also check the above reference for detailed overview of **Oracle's approach to B-Tree implementation for indexes** and especially look at example/figure 3-1 (copied/pasted below for quick indication) in the referenced document. Also, visit http://use-the-index-luke.com/sql/anatomy/the-tree for another reference.



Sample Dataset / Assumption:

Data in hard-disks is stored in the form of data blocks whereas Operating Systems manage records/data storage in the form of pages. Due to security restrictions, we cannot have access to block numbers / page numbers to be stored in B-Tree for quick reference. Therefore, we will make few assumptions for this project. Assume, data/records (sample given below) are stored line by line (L1, L2, L3, …) in different files (F1, F2, F3, …) saved on different hard drives/partitions (C, D, E, etc.). First record/row in file F1 on C drive may be referred as CF1L1 and an entry in leaf node of index/B-Tree for student id 0 may refer to CF1L1 for quick access. There can be multiple/different indexes on one table for different columns e.g. separate indexes for Name, Gender, Student Id, DOB etc.

St_ID	Name	Father	DoB	M/F	Reg Date	Reg Status	Degree Status	Address	Qualification
0	Hussain Ansary	Ubaid Ansary	5-Jan-74	M	13-Aug-94	A	C	h# 978 Street No.72 Defense Phase 1 KHR	A-Level
1	Shk. Munir Hussaini	Viqar Hamid Hussaini	13-Dec-74	M	13-Aug-94	A	C	H# 169 s# 0 Mutian wala Thata	A-Level

Detailed sample dataset is present in compressed folder named "DatasetFall2022DSproject"

Requirements:

You are required to develop an **application** with features / functionality / options listed below:

1. Creation of new index/B-Tree for a particular column/attribute.
2. Option for node insertion, deletion, and search. Creation of new record will result into insertion of new record/line in a file plus node/entry in index, deletion of record will result in deletion of record from file as well as deletion of node from B-Tree, and search related queries will use index for efficient search of data/record from hard drive.
3. Leaf nodes will be having reference to records location whereas non-leaf/internal nodes will be used just for search/classification of keys (Reference Oracle's approach).
4. Index/indexes is/are also required to be stored / saved on hard drive. Each node should be stored in a separate file and addressing of nodes from one level to next should be dynamic i.e. while accessing an index, root will be loaded into

memory/RAM. In case of multi-level index, next node from 2nd level will be loaded (2nd disc access) after searching concerned path from root node.

5. Queries/questions on the basis of indexed column should use concerned index. E.g. a question like "Number of male students" should use index on gender column, if an index is there.
6. Application should support joins (AND), union (OR), Minus (NOT), value-based and range-based queries.
7. Application should be menu driven.

Project Part 2

Instructions:

In this part of the project you have to play with the **Graph** data structures .You are provided with a file named (friends) containing a **Social Network** (friendship relationships) among friends. This file corresponds to the file "Fall2022DSDataFile008.txt" originally provided to you and captures the friendship between profiles present in file (Fall2022DSDataFile008.txt): profile ID 2001 to profile ID 2500. Each row contains the profile ID staring from 2001 and then a list of profile ID's as friends of the former. Note that friendship is commutative so if A is a friend of B, B is also a friend of A. Once you have the graph of the friendships you are required to complete the following tasks or handle the following queries:

Terminology:

The **social circle** of a person is all the persons that the person is friends with, and all the persons that they are in turn friend with. For example if A is friends with only B, and B has D, G and Z as friends, then A's social circle includes B, D, G and Z.
Social distance between two profiles is the number of intermediary profiles. Two friends have a social distance of zero. A friend of friend is 1 social distance away and so on. So all the profiles in the social circle of a person are at most 1 social distance away.

Tasks:

1. Find the person with the biggest social circle.

2. **Bonus Task:** Let’s say two people have a chance to meet each other if their social circles intersect. For the provided dataset are there any two students who will never meet? List all such pairs if they exist.

Queries:

1. For any two input profiles find the social distance between them.
2. For any two given profiles find the common profiles in their social circles