# INF-2700

# Assignment 2


# Magnus Dahl-Hansen


# October 2022

## Introduction:

This report describes how a given database management system has been improved to support queries that is not restricted to equality and to search for elements using binary search instead of linear search. The report also includes a discussion between the performance of linear search VS binary search and how a B+-tree affects the storage size and performance of queries.

## Compiling and running the program:

The program is compiled by navigating to the folder "*mda105/assignments-2-3/db2700*" in the terminal and run the "make" command.

This will generate two executable files which can be run by typing *"./run_test"* to run the tests used to compare linear search and binary search and "*./run_front"* to use the base program. While using the base program the user can type the "*help"* command to understand how the program works.

The command "*make cleanall*" will clean all the generated files and restore the folder back to it's original state.

## Design & implementation:

The given base program only supported queries with equality search by using the logical "*equal*" operator to compare an element from the database with a search element. If the two elements are equal then the element from the database is placed inside a new schema and displayed for the user. The base program have been extended to support queries with the: "*less than"*, "*less than or equal*", "*bigger than*", "*bigger than or equal*" and "*not equal*" operations by using the same logic. The logical operators: " < ", " <= ", " > ", " >= " and " != " are used to compare two elements and place them inside a new schema based on whether or not the operator return a True or False value.

The base program have been upgraded to support binary search for equal elements as well as linear search. A binary search is an algorithm that finds a specific element in a sorted array. It works by dividing the array in two halves, see which half that might contain the element based on it's size and discard the other half that can not contain the element. This process is repeated until there is only one element left in the array.

This algorithm is used to find a specific element among the blocks that the database consists of. The binary algorithm is applied to the database management system by fetching the total number of blocks that the database consists of and dividing them by two to find the middle block. A

comparison test is then performed to see if the search value is less or greater than the first record in the middle block.

In a case where the first record is less than or equal the search value then a linear search through the middle block is performed. If the search value is not found in this page then the lower half of the blocks are discarded and the process is repeated until the record is found. Should however the first record in the middle block be greater than the value that is searched for the greater half of the blocks are discarded and the process repeats.

If a record is found then a new schema is created, the record is copied to the schema and displayed to the user. If the value is not found a error message is displayed.

To compare the linear search to the binary search four tests have been implemented that fetches different records from a database consisting of 1000 elements. A linear and a binary search is performed on an equal value and the number of disk seeks, reads, writes and IO's are recorded. The test results are then printed to the terminal for the user to analyze.

A minor mention is that to make it easier to test the database program, a function has been implemented that will create a table with 1000 integer records. This table is called "*table*" and can be accessed upon running the "*./run_front*" program.

## Performance (Linear VS Binary):

The tests that evaluate the performance of linear and binary searches have been run with a database that contains 1000 sorted integer records ranging from 0-999. Four tests have been executed with the search values: 0, 59, 60 and 999. The test results can be viewed on the next page.

The tests shows that linear searches are most optimal when searching for values between 0-59 which are located in the five earliest database blocks. A linear search has to read from 1-5 blocks to locate these values while a binary search has to read 7 blocks to locate these values. However when the search values are greater than 59 (meaning they are located from block number 6 and up) binary searches are most optimal. The binary search needs to read 7 blocks on average to find a value while a linear search needs to read 43 blocks on average to find a value in a database consisting of 84 blocks.

```
TESTING LINEAR SEARCH with value: 0
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 1/1/0/1
INFO:  test_tbl_read() succeeds.
DONE
TESTING BINARY SEARCH with value: 0
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 6/7/0/7
INFO:  test_tbl_read() succeeds.
DONE

TESTING LINEAR SEARCH with value: 59
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 1/5/0/5
INFO:  test_tbl_read() succeeds.
DONE
TESTING BINARY SEARCH with value: 59
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 7/7/0/7
INFO:  test_tbl_read() succeeds.
DONE

TESTING LINEAR SEARCH with value: 60
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 1/6/0/6
INFO:  test_tbl_read() succeeds.
DONE
TESTING BINARY SEARCH with value: 60
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 5/5/0/5
INFO:  test_tbl_read() succeeds.
DONE

TESTING LINEAR SEARCH with value: 999
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 1/84/0/84
INFO:  test_tbl_read() succeeds.
DONE
TESTING BINARY SEARCH with value: 999
INFO:  test_tbl_read ("Me") ...
Num blocks in schema: 84
INFO:  Number of disk seeks/reads/writes/IOs: 6/7/0/7
INFO:  test_tbl_read() succeeds.
DONE
```

***Picture 1. Test results***

## B+ tree discussion:

The main difference between a binary search tree and a B+ search tree is that the B+ tree can have more keys and child nodes than a binary search tree. This means that the lookup time for a node or an element is greater in a binary tree because of its height and that there are more nodes to go through. For this reason, a binary tree is better to use in smaller databases stored in fast memory such as RAM. A B+ tree should however be used in larger databases stored in a slow type of memory such as a disk, because a B+ tree can significantly reduce the lookup time since it has less accesses than a binary tree.

A B+ tree not only contains nodes as a binary tree does, but keys for each node as well. For this reason a B+ tree will use more memory space than a binary tree which is another reason why it should be used in larger databases while binary trees should be used in small databases.

As a conclusion to this topic, the B+ tree will take up more space and may cause a longer lookup time for elements in this database because it is small. However it could also boost the performance if the database were stored on a disk.