University of Tromsø

Department of computer science

INF-2700: Database system concept

Assignment 2
DBMS elements

Camilla Stormoen, cst042

October 20, 2015

1 Contents

2 Introduction

This report describes the implementation of some basic elements of a database management system (DBMS). One element is to extend the base program such as queries on integer attributes are not restricted to equality search and the other is to extend the base program to use binary search on an integer field.

3 Technical Background

3.1 Linear Search

A linear search looks down on a list, one item at a time until the element is found or the list is exhausted. The list does not need to be sorted. Linear search complexity is O(n) search. [1]

3.2 Binary Search

A binary search algorithm finds the position of a target value within a sorted array. The algorithm begins comparing the target value of the middle element of the sorted array. If the target is equal, the position is returned and the search is finished. If the target is less than the middle value, the search continues on the lower half of the array, or if the target is greater than the middle, the search continues on the upper half of the array. The process continues to the target is found. Average case performance is O(log$n$). [2]

4 Design

4.1 Extend the type of queries

The base program supports only equality search on integer attributes, but should also support searches like *<, <=, >, >=* and *!=*. To determine which operator to use, a string compare can check for similar o*p* (what user type in) with the different operators and then return the result of the operation.

4.2 Binary Search

The base program supports linear search, but should support binary search on an integer field. When finding the value and record in the database system, different calculations must be done. First, the block number must be found. Finding the block number needs to calculations, one that is the number of records per block and the other the middle of the records. When the block number is found, it is used to get the page from the block and put in memory.

Then some calculations are made to find where to start reading the value. The calculation get the record integer value that is used to compare with the input value from user. If the values are equal, the position is set, there's a page validation for the position and then the page record is returned.

If the middle value is less than the input value, the search must continue in the lower half and gets repeated to the right value is found. If the middle value is greater, the search continues in the upper half to the right value is found. All this is shown in figure 1 below.

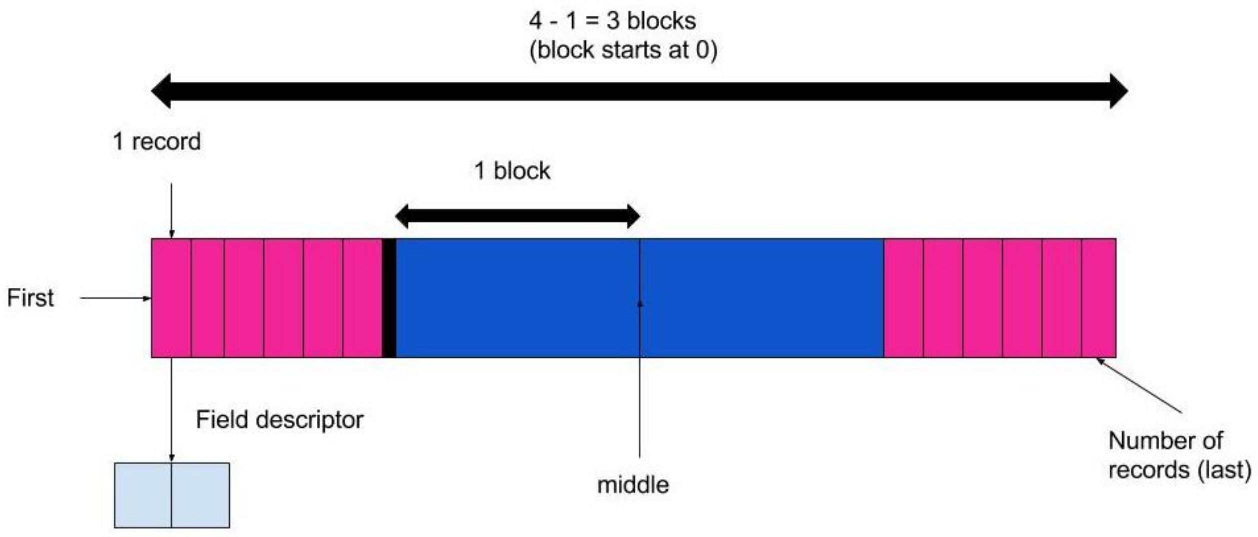

Figure 1 *Show the design of the binary search and how the blocks, pages and field descriptors are connected.*

5 Implementation

The implementation is written in C language and executed using the Linux terminal on the ifilab computers.

The performance evaluation of this system is used by the given profiling capability in the base program.

6 Discussion

6.1 Linear search vs binary search

Linear search takes long time, O(n) and as we can see from figure 2, the longer out in the list the element is, the longer time, more reads and more IO is used by the program. The disk seek is so low, shifting between 1 and 2, they are almost invisible in the graph. The highest number of disk reads and IO is here almost 16000 times. The graph tells us that if the id is small, like the first one 50, number of disk reads, seek and IO are not high, but since the list is sorted, many elements must be checked before the value is found, which results in a high number of reads and IO.

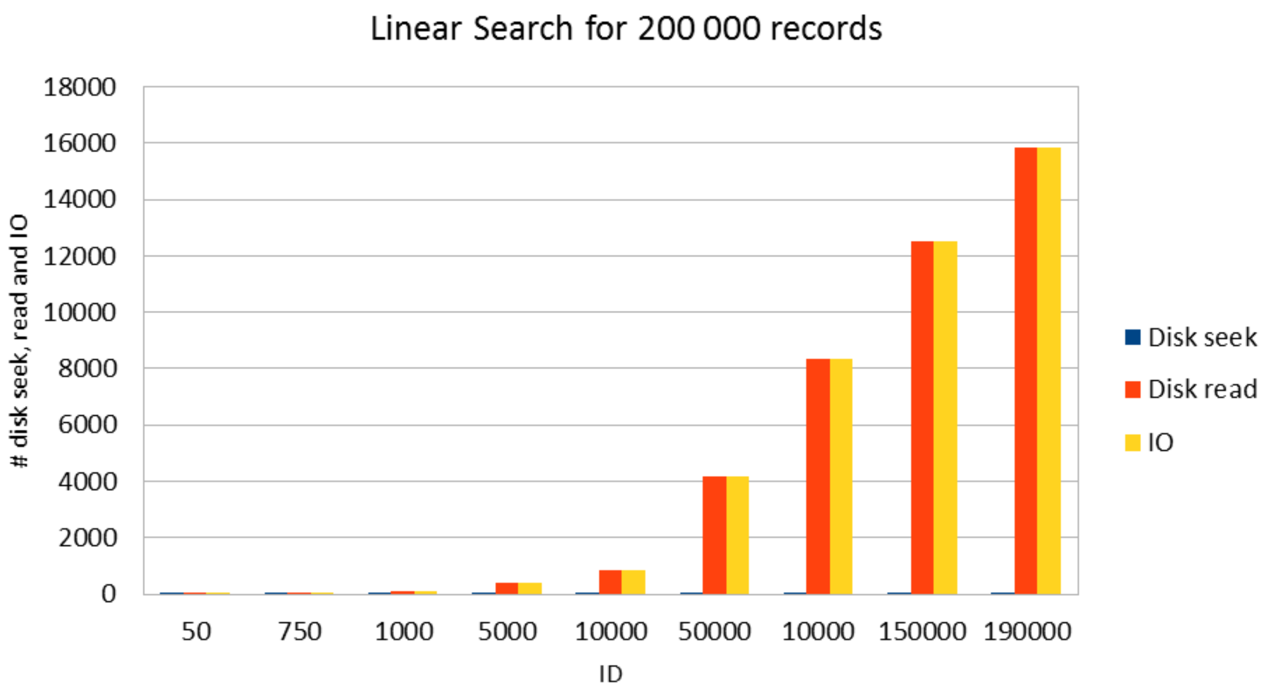

Figure 2 *Show how many disk seek, read and IO that needs when reading an ID from a database.*

As in figure 3, the results from numbers of disk reads, seek and IO is much more even. The highest number of disk reads and IO is here 16 and 14, and here are also the disk seek much higher than in the linear search. The graph shows that it takes almost the same time, it's only with ID 150 000 that disk seek, read and IO goes down drastically.

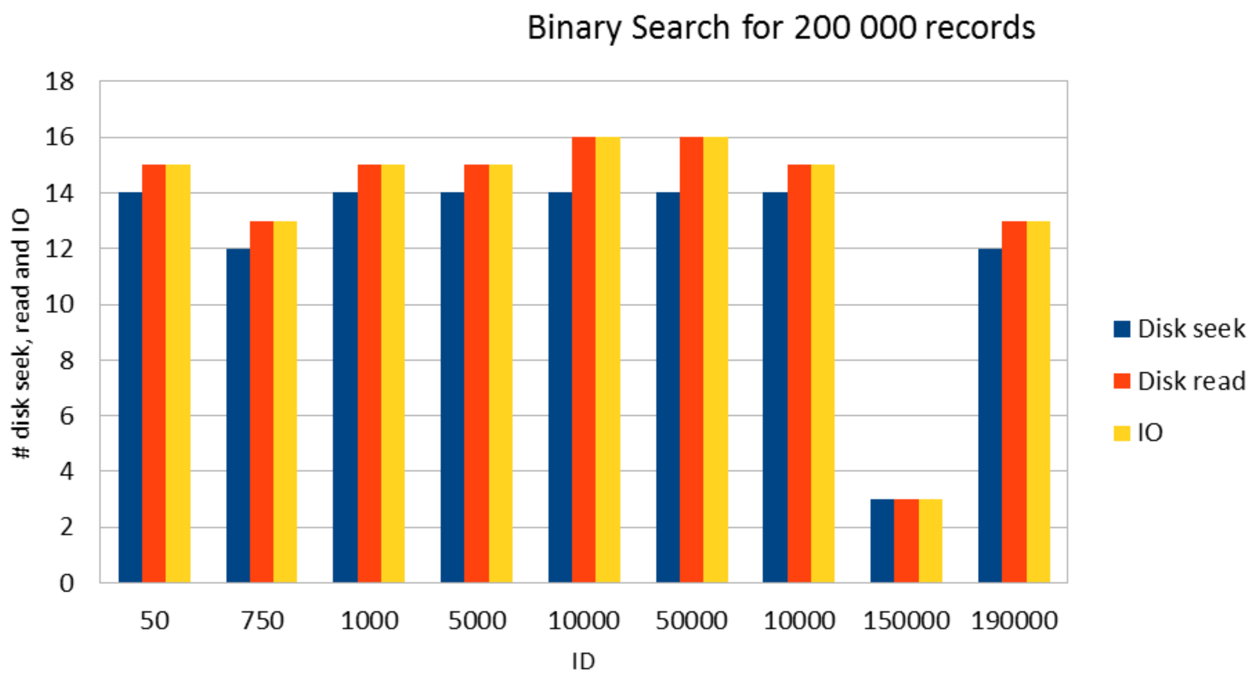

Figure 3 *Show how many disk seek, read and IO that needs when reading an ID from a database.*

As we can see from the two graphs, is that binary search is much more effective when searching. When having a sorted list, can finding a small element in the beginning of a list be effective, but when accessing greater elements, linear search use to many disk seeks, reads and IO. Here is where binary search is much better by always use the middle value to check for comparison. The only downside of a binary search is that the list must be sorted before searching. A linear search does not have to be sorted. If an unsorted list is being linear search and finds the element in a short time, it is most likely to be a lucky hit. All things considered, a binary search is therefor better.

7 Conclusion

This report has described the implementation of some basic elements of a database management system (DBMS). One element is to extend the base program such as queries on integer attributes are not restricted to equality search and the other is to extend the base program to use binary search on an integer field. As we can see from the results from the graphs, binary search is much better in a database management system.

8 References

[1] W. contributors, «Linear search,» Wikipedia, The Free Encyclopedia, 10 October 2015. [Internett]. Available: https://en.wikipedia.org/w/index.php?title=Linear_search&oldid=685026770. [Funnet 20 October 2015].

[2] W. contributors, «Binary search algorithm,» Wikipedia, The Free Encyclopedia., 20 October 2015. [Internett]. Available: https://en.wikipedia.org/w/index.php?title=Binary_search_algorithm&oldid=686598231. [Funnet 20 October 2015].