Midterm Sample Question and Response

CS165-Staff

1 Question: Comparing B-Tree Search Costs and Binary Search costs

This example is taken from class 9 and gives an idea of how we expect you to answer questions on the quiz. The answers should be a mixture of textual descriptions and algebraic modeling. It differs only in that the index in class 9 is assumed to be clustered, whereas the ones in this example are unclustered.

Note: The actual midterm shall consist of many of these questions and be significantly longer. You should not only be able to produce the answers below but be able to produce them quickly.

Your goal is to run a select query over a column of integers: table1.col1. For the purposes of this select we will perform the operator in two different ways. In the first case, we will assume we have an unclustered B-Tree over table1.col1 and do the predicate evaluation by using our B-Tree. In the second case, we will assume that we have an unclustered sorted copy of column table1.col1 as an index.

As parameters for the cost models, let the input size be n elements, the page size be k, and assume the system is a 64-bit operating system. You may assume B-tree nodes are a single page in size, and that integers are 64 bits.

a) Assume the select query is an equality predicate of the form table1.col1 $= x$. Assume that there are no more than a handful of values in table1.col1 with the value. Compare the costs to perform the select operator using a B-Tree vs. using a sorted array.

b) Now assume we are searching for a range of values $x1 <$ table1.col1 $< x2$, and let the percentage of qualifying tuples be given the label s. Model how the costs given above would change in terms of data movement as s changes. Furthermore, describe how data access patterns might influence whether it would be faster to use a sorted array or a B-Tree. [5 sentences max]

2 Answer

a) We will perform binary search over a sorted array of $< val, pos >$ pairs. At each iteration i of our binary search we will make a jump of $\frac{(8+8)n}{2^i}$ elements. We will access a

new page each time this value is greater than a page size, so we can make a conservative approximation to the number of pages touched by solving $\frac{16n}{2^i} = k$ for i. Solving this equation gives: $i = log_2 \frac{16n}{k}$. Since the equality predicate is assumed to have only a handful of qualifying values, then (If we plug in the numbers from class of n=1,000,000,000 and k = 64,000 then we get our class answer of 17 pages)

For the B-Tree, ideally we would like each node to fill a page's worth of memory. Each value takes 16 bytes, 8 for the integer and 8 for the pointer, and so a node will hold a maximum of $\frac{k}{16}$ values. Since each node only needs to be half full, we can assume each node holds at least $\frac{k}{32}$ values. Thus a conservative estimate of the number of pages touched is $log_{\frac{k}{32}} n$, where an aggressive estimate would be $log_{\frac{k}{16}} n$. (For our class example, using either estimate produces that the B-Tree requires 2 pages for searching for a value).

b) From a modeling of data movement perspective, we find that both the models given above will have an added cost of $\frac{16n * \frac{s}{100}}{k}$. As s grows, this will be the dominant cost for both models. Additionally, the B-Tree stores this data fragmented in memory, compared with a sorted array which stores this data contiguously. Thus jumping between leaf nodes will cause some overhead beyond just looking at the data and we would expect the model to underestimate the cost of a B-Tree search compared to the binary search over the sorted array.