COM S 363 HW2

prolling

February 2023

Suppose I have a relation Grades(student_id, assignment_id, score). I have 150 students and 20 assignments. I would grade all submissions of one assignment based on the submission order and then insert the records. As a result, based on my insertion nature, this relation is not sorted on student_id but sorted on the assignment_id. I choose heap file as my file organization. My page is quite small – it can only store 20 records or 200 bytes on one page. The SearchKeySize is 4 bytes and PointerSize is 2 bytes. My buffer size is also small, 5 pages.

Question 1

(50 points) If my most frequent query is to find grades of individual students (e.g., select assignment_id, score from grades where student_id='3347';)

a.

(5pts) What is the average I/O cost (i.e., number of pages in terms of reading and writing) for this query if I **don't build index** for student_id? (note: **student_id can appear many times** in this relation)

The number of records is 150 students * 20 assignments. The number of records that can fit on a single page is 20, making the total number of pages 150. Since it is not sorted, we must scan through every page, which is 150 pages.

ANSWER = 150 pages

b.

I want to improve the I/O cost. I am debating if I need to build index for student_id, or to sort based on student_id. So I need to do some estimation. Please help me by answering the following questions.

i.

(15pts) What is the I/O cost of multi-way merge sort (aka, external sort) if I sort the relation after I enter all records? Explain the process.

I/O Cost for Multi-Way Merge Sort = $2N + 2N(\lceil log_{B-1}N/B \rceil)$

The process includes:
1. Taking the readable amount of files 2. Sorting those files 3. continuing this process and merging the sorted sub-groups

The parts of the equation mean:
a. log_{B-1} = the merging
b. N/B = the number of subfiles
c. B = available size for memory
d. N = total pages
e. $\lceil\rceil$ = make it a whole number because you can't read or write 0.1 pages

In our case:
N = 150
B = 5
I/O Cost = $2(150) + 2(150)(\lceil log_{5-1}150/5 \rceil)$
I/O Cost = 1200

ANSWER = 1200

ii.

(15pts) Suppose I decide to build B+ tree index instead of sorting. What is the smallest number of pages do you estimate the B+ tree will take?

There are 3000 data nodes (150 students * 20 assignments each)
Number of Pointers = n
n * SearchKeySize + (n+1) * PointerSize $\leq$ PageSize
n * 4 + (n+1) * 2 $\leq$ 200
4n + 2n + 2 $\leq$ 200
6n $\leq$ 198
n $\leq$ 33
n = 34

Number of Pointers or fanout(max) = 34
One pointer will need to be reserved in order to point to the next data entry node.

Since the max fanout is 34:
$\lceil 3000/33 \rceil$ = 91
$\lceil 91/34 \rceil$ = 3

Here is a simple visualization of the B+ tree:
root node
3 directory nodes
91 data entry nodes
3000 records (not counted in the number of pages)

```
          -
          - - -
     ——....——-
———-............————
```

umber of pages = 1 root node + 3 index nodes + 91 pointers = 95 pages

ANSWER = 95 pages

iii.

(15pts) What is the worst I/O cost for answering this query with B+ tree index now?

The worst case I/O cost for answering this query is 93 because the worst case is that it must read each record and also the parent and index node.

ANSWER = 93

Question 2

(40 points) If my most frequent query is to find all scores for an assignment (e.g., select student_id, score from grades where assignment_id='10';)

a.

(10pts) What is the average I/O cost if I don't build index for assignment_id? (note: **assignment_id is sorted** and each assignment_id **can appear as many as 150 times** in this relation)

We have 8 pages because it can appear as many as 150 times and $\lceil 150/20 \rceil = 8$

The I/O cost can be calculated as follows:
$\lceil log_2 150 \rceil + 8$

ANSWER = 16

b.

I wonder if building an index for assignment_id would further improve the I/O cost. Please help me by answering the following questions.

i.

(15pts) Suppose I decide to build B+ tree index. What is the smallest number of pages you estimate the B+ tree will take?

on calculations before, our maxfanout is 34, which is 33 after taking into account that one pointer will need to be used to point to the next one.
We have 150 pages. This means that we need to do $\lceil 150/33 \rceil = 5$
Now we need to add one to this for the root. This gives us a total of 6 pages.

ANSWER = 6

ii.

(15pts) What is the best I/O cost for answering this query with B+ tree index now?

The best case of the I/O cost would be 3 because the best case is that there are not appearing 150 times. That would mean that you would only need to read the root, the index node, and the page that the record is on. However, if it did appear 150 times, which is possible according to the problem specifications above, then you must read an additional 7 pages, which would result in 10 pages total. Therefore, the best case is 3 pages.

ANSWER = 3

Question 3

(10 points) Suppose at the end of the semester, I need to curve the grades. I decide to increase all scores by 5 points. What is the I/O cost for this operation? Will my index on student_id be helpful in reducing I/O cost for this operation?

The I/O cost for this operation is equal to the number of scores times 2. The number of scores is 150 pages and this * 2 = 300 pages. The index on student_id would not be helpful because each score needs to be read anyways. It would also be unhelpful as having an index and writing to it results in a higher I/O cost than unindexed as the indexed has to reorganize.