Khushveen Kaur Umra

COMS 363

10th October 2021

Homework 2

Suppose I have a relation Grades(student_id, assignment_id, score). I have 200 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, the student_id is not sorted, but the assignment_id is. I choose heap file as my file organization. My page is quite small – it can only store 40 records, or 200 bytes in one page. The SearchKeySize is 2 bytes and PointerSize is 2 bytes. My buffer size is also small, 4 pages.

1. **(50 points) If my most frequent query is to find individual students, such as select * from grades where student_id='3347';**

   a. **(5pts) What is the 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 as many as 20 times in this relation)**

➔ We have 200 students and 20 assignments.
Hence, the number of records can be calculated as:

Number of records = number of students * number of assignments
= 200 * 20
= 4000.

Hence, we have 4000 records.

Now, the number of pages can be calculated as:

Number of pages = Number of records / Number of records a page can hold
= 4000 / 40
= 100.

Therefore, the number of pages in terms of reading and writing for this query if we don't build and index for student_id is 100 pages, and it needs to read the entire relation.

   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.**

➔ The first phase would be 'partitioning'. So, the I/O cost will be:

i) For reading = 100
ii) For writing = 100

Total: I/O = 200 (We will get 25 subfiles)

The second phase will include three rounds. In the first round, you will have to merge every 3 subfiles into one.

Then, the I/O cost would be: - For reading = 100

- For writing = 100

Total I/O = 200 . Now we will get 9 subfiles.

Now in the second round, you will again merge every 3 subfiles into one.

Then, the I/O cost would be: - For reading = 100

- For writing = 100

Total I/O = 200 . Now we will get 3 subfiles.

Now, for the final third round, we will again merge every 3 subfiles into one.

Then, the I/O cost would be: - For reading = 100

- For writing = 100

Total I/O = 200 .

■ **Here the process has been completed.**
And the final I/O cost of multi-way merge will be; 200+200+200+200 = 800.

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?**

➔ Using the formula; n* SerachKeySize + (n+1) * PointerSize <= PageSize

n * 2 + (n+1) * 2 <= 200
2n + 2n + 2 <= 200
4n <= 198

Therefore, n <= 49.5 ; n = 49 & n+1 = 50.
Hence, each page can store 49 keys and 50 pointers.

Now, as previously calculated, we have 4000 records. And in a dense index method, we have one pointer per record.

Then, on the leaf level in the B+ tree index, we will have: 4000 pointers
Which means, we will have, 4000 / 49 = **82 pages on the leaf level.**

Now, the B+ tree index has three- four levels.

Hence, the smallest number of pages that the B+ tree will take is:

82 + 2+ 1 = 85 pages.

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

➔ We have 20 assignments to read, hence using the B+ tree index, we will have to read 20 data pages.

Now, the B+ tree index has three or four levels. So, the tree index will need/read three or four index pages.

Therefore, the worst I/O cost for answering this query with B+ tree index would be either;

20 + 3 = 23 pages or 20 + 4 = 24 pages.

2. **(40 points) If my most frequent query is to find all scores for an assignment, such as**
   select * from grades where assignment_id='01';
   a. **(10pts) What is the 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 200 times in this relation)**

➔ We have 100 pages.
Hence, for using the formula for binary search, we get;
Log_2 (100) = 6 or 7 pages

Now, a page can hold up to 40 records.
Therefore, to retrieve all the records; 200 / 40 = 5 pages.

Therefore, the total I/O cost if we don't build index for assignment_id is;

6 + 5 = 11 pages or 7 + 5 = 12 pages

b. I am debating if building 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 do you estimate the B+ tree will take?

➔ Now, in a sparse index method, we have one pointer per page in the B+ tree index.
Since we calculated that we have 100 pages,
On the leaf level of the B+ tree index, we will have 100 pointers.

Then, the numbers of pages on the leaf level will be; 100 / 49 = 3 pages.

Here, the three have two levels. So, the smallest number of pages the B+ tree will take is;
3 + 1 = 4 pages.

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

➔ Here, the B+ tree index will read about 5 ± 1 data pages and it will read about two or three index pages.
Therefore,
The best I/O cost for answering this query with B+ tree index now will be either, 6, 7, 8, or 9.

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?