CS W186 Databases - Fall 2019
Guerilla Section 2: Sorting, Hashing, and Buffer Management

Sunday, September 29, 2019

1 Sorting and Hashing

Suppose the size of a page is 4KB, and the size of the memory buffer is 1 MB (1024 KB).

1. We have a relation of size 800 KB. How many page IOs are required to sort this relation?

   Answer: **400**
   200 to read in, 200 to write out. Since the relation is small enough to completely fit into the buffer we only need to read it in, sort it (no I/Os required for sorting), then write the sorted pages back to disc.

2. We have a relation of size 5000 KB. How many page IOs are required to sort this relation?

   Answer: **5000.** 2 passes. 5000 KB with 4KB per page means 1250 pages are needed to store the relation. We have 1024 / 4 = 256 pages in our buffer. Number of Passes $= 1 + \lceil \log_{255} \lceil 1250/256 \rceil \rceil = 2$

3. What is the size of the largest relation that would need two passes to sort?

   Answer: **261,120 KB.** (255 * 256 pages).

4. What is the size of the largest relation we can possibly hash in two passes (i.e. with just one partitioning phase)?

   Answer: **261,120 KB.**

5. Suppose we have a relation of size 3000 KB. We are executing a `DISTINCT` query on a column `age`, which has only two distinct values, evenly distributed. Would sorting or hashing be better here, and why?

   Answer: **Hashing**, which allows us to remove duplicates early on and potentially improve performance (in this case, we might be able to finish in 1 pass, instead of 2 for sorting).

6. Now suppose we were executing a `GROUP BY` instead. Would sorting or hashing be better here, and why?

   Answer: **Sorting** because hashing won't work; each partition is larger than memory, so no amount of hash partitioning will suffice.

2 External Sorting

Assume our buffer pool has 8 frames. In this question, we'll externally sort a 500 page file.

1. How many passes will it take to sort this file?

   Answer: **4 passes.** Number of Passes = $1 + \lceil \log_7 \lceil 500/8 \rceil \rceil = 4$

2. Given the number of passes you calculated in 2.1, how many I/Os are necessary to externally sort the file?

   Answer: **4000 I/Os.** 2 * Number of Pages * Passes = 2 * 500 * 4 = 4000 I/Os.

3. What is the minimum number of additional frames needed to reduce the number of passes found in 2.1 by 1?

   Answer: **1 additional frame.** Given that we had 4 passes in 2.1, we need to calculate how many pages it will take to sort the relation in 3 passes.
   $B(B-1)^2 >= 500$
   B = 9 frames. 9 - 8 = 1 additional frame. I/Os.

4. What is the minimum number of additional frames needed to sort the file in one pass?

   Answer: **492 pages.** If we can fit the entire table into the buffer, our initial sorting pass will sort the table. Therefore 500 - 8 = 492 pages.

3 Buffer Management

We're given a buffer pool with 4 pages, which is empty to begin with. Answer the following questions given this access pattern:

A B C D E B A D C A E C

1. What is the hit rate for MRU?

   Answer: **4/12.**

A						*	D				
	B				*						
		C						*	A		
			D	E						*	C

2. In order of when they were first placed into the buffer pool, what pages remain in the buffer pool after this sequence of accesses?

   Answer: **B, D, A, C**

3. What is the hit rate for clock?

   Answer: **5/12.**

4. Which pages are in the buffer pool after this sequences of accesses?

   Answer: **A, C, D, E.**

5. Which pages have their reference bits set?

   Answer: **A, C, E.**

6. Which page is the arm of the clock pointing to?

   Answer: **A. On a page hit, the arm doesn't move.**

4 Hints for Question 1: Hashing and Sorting

1. Convert all numbers to pages, and then apply the equation you learned in class.

2. Same.

3. Again, convert all numbers to pages.
The second pass merges how many runs?
For the second pass to complete the sort, this must be enough to merge all the runs created by the first sort. How big does the relation have to be for the first sort to create that many runs?

4. Similar to the previous question, but think about the hashing protocol instead.

5. Which protocol benefits from removing duplicates?

6. `GROUP BY` does not allow us to remove duplicates. Are both protocols capable of handling such large numbers of duplicates?