CS150A Quiz 4 Solutions

External Hashing and Sorting

Q1: How many passes are required to fully sort a 8192 MB file with external merge sort? (+2 points for correct)

We'll start by figuring out how many pages our buffer contains and how many pages the file contains.

We perform the following calculation to find the number of buffer pages we have allocated:
B = # buffer pages = 32 MB * (1024 KB / 1 MB) * (1 page / 128 KB) = 256 pages.

We perform the following calculation to find the number of pages in the file:
N = # pages in file = 8192 MB * (1024 KB / 1 MB) * (1 page / 128 KB) = 65,536 pages.

Let's use the formula for number of passes in an external merge sort: $\lceil \log_{B-1} \lceil \frac{N}{B} \rceil \rceil + 1$.
Plugging in N = 65536 pages and B = 256 pages, we see that **3 passes are required**.

Q2: What's the I/O cost of fully sorting a 8192 MB file with external merge sort? (+2 points for correct)

The I/O cost follows directly from the number of passes it takes to sort the file. In particular, it will take 2 * (# of pages in the file) * (# of passes).

Hence, we get 2 * 65536 * 3 = **393,216 I/Os**

Q3: Suppose we double the size of our buffer, to 64 MB. What is the largest file size (in MB) that we can externally sort in two passes? (+2 points for correct)

In this scenario, B = 512. B*(B-1) = 512 * 511 = 261,632 pages = **32,704 MB**.

Q4: Generalizing Q3, if we double the size of our buffer, approximately how much larger of a file can we externally sort in k passes? (+2 points for correct)

The largest file we can sort in 2 passes is B*(B-1). The largest file we can sort in *k* passes is B*(B-1)$^{k-1}$ (can you reason why?). Thus, doubling B will increase the max file size for *k* passes by a factor of **2^k**.

Q5: You decide to separate your dataset with external hashing. How does the I/O cost of externally hashing the file compare with the I/O cost of externally merge sorting the file? (+2 points for correct)

Assume that the data is uniformly distributed on the hashed key and that your hashing function distributes the records into partitions evenly.

Given the conditions presented for external hashing, in particular the perfectly distributed hash, we see that it will take 3 passes to externally hash the whole file. This results in **the same I/O cost**.

To verify this, let's step through this pass by pass. After pass one, we have 255 partitions, each at least 257 pages large. This requires recursive partitioning on all partitions - another complete pass. At this point we would conquer each partition, which is a final and third pass.

Q6: Suppose you are hashing a file and one of the partitions is 36 MB after the first pass (all other partitions can fit in the 32 MB buffer). How much larger is the I/O cost of externally hashing this file, compared to a scenario (with the same file) in which no partitions are ever oversized? (+2 points for correct)

Assume that a new hashing function is chosen for the second pass such that the records are distributed in a way that guarantees subsequent partitions to be under 32 MB.

Compared to a scenario with no oversized partitions, we need to recursively partition the 36 MB partition. Note that we do not need to do anything to the other partitions before conquering, and the IO cost of the conquer pass will not change - it does not depend on the number of partitions.

Thus, we take a single pass over 36 * 1024 / 128 = 288 pages, which is 2 * (288) = **576 I/Os** (a pass involves reading and writing each page once - 2 I/Os per page).

Relational Algebra

$$A)\ \pi_{sname}(\sigma_{\text{color} = \text{'pink'}}(Reserves \bowtie Boats) \bowtie Sailors)$$
$$B)\ \pi_{sname}(\pi_{sid}(\sigma_{\text{color} = \text{'pink'}}(Reserves \bowtie Boats)) \bowtie Sailors)$$
$$C)\ \pi_{sname}(\sigma_{\text{color} = \text{'pink'}}(Reserves \bowtie Sailors) \bowtie Boats)$$
$$D)\ \pi_{sname}(\sigma_{\text{color} = \text{'pink'}}(Reserves \bowtie Boats \bowtie Sailors))$$
$$E)\ \sigma_{\text{color} = \text{'pink'}}(\pi_{sname}(Reserves \bowtie Boats) \bowtie Sailors)$$
$$F)\ \sigma_{\text{color} = \text{'pink'}}(\pi_{sname}(Reserves \bowtie Sailors) \bowtie Boats)$$
$$G)\ \sigma_{\text{color} = \text{'pink'}}(\pi_{sname}(Reserves \bowtie Boats \bowtie Sailors))$$

Q7: Which of the relational algebra(s) above describe(s) the name of all sailors who have reserved pink boats?

ABD

C's first join wont work because there is no column color

EFG materializes too early and do not have the color column by the time of selection (also in general queries need to end in projections)

Q8: Which one of the above expressions that is correct, if executed as a query plan, is the most performant?

B

Since B has selected only the ids out for the join (as opposed to carrying around extra information not needed) AND it doesn't do the three way join but only joins once the filter is applied.

$$\rho(temp, \pi_{bid}(\sigma_{\text{color} = \text{'blue'}}\ Boats) \cap \pi_{bid}(\sigma_{r\_date \geq \text{'2016-01-01'}}(Reserves \bowtie Boats)))$$
$$\rho(result1, \pi_{sname}(Reserves \bowtie temp \bowtie Sailors))$$
$$\rho(temp1, \pi_{sid}(\sigma_{\text{color} = \text{'blue'}}(Reserves \bowtie Boats)$$
$$\rho(temp2, \pi_{sid}(\sigma_{r\_date \geq \text{'2016-01-01'}}(Reserves \bowtie Boats)))$$
$$\rho(result2, \pi_{sname}(temp1 \cap temp2 \bowtie Sailors))$$
$$result1 - result2$$

Q9: Which of the following *could* be in the output?
 sailors who reserved blue boats but no other boats
 sailors who reserved boats in 2016 but only blue boats
 sailors who have only reserved boats after 2016
 sailors who have reserved blue boats that were reserved by others in 2016 but not the sailor him/herself in 2016

1,4

result1 first finds all the blue boats reserved after 2016, then finds the sailors who have ever reserved those boats, even in 2015.
result2 finds sailors who have reserved blue boats anytime in the past, and reserved boats in 2016. So if a sailer has only reserved pink boats, then the sailor will not be included.

A student made a great private post that explains each choice in detail and chose to stay anonymous; here is the answer (with slight modification):

*Since we want to determine if a record *could* be in the result. Then the answer should be true iff exists a specific record satisfying certain conditions that is in result1 but not in result2.*

1. sailors who reserved blue boats but no other boats
   YES: Consider a sailor S who only made one reservation ever, in which he reserved a blue boat B in 2015. The boat B is reserved again by another sailor in 2016.
   $B \in temp$ □ $S \in result1$. $S \notin temp2$ □ $S \notin result2$
2. sailors who reserved boats in 2016 but only blue boats
   NO: Obviously, any such sailor will have his/her record in result2, which prevents it from being in the result.
3. sailors who have only reserved boats after 2016
   NO: If a record of a such sailor S actually is in result, S can't be in result2. Then S must have not reserved any blue boat, which prevents his/her record from being in result1. Contradiction.
4. sailors who have reserved blue boats that were reserved by others in 2016 but not the sailor him/herself in 2016
   YES: Again, consider a sailor S who only made one reservation ever, in which she reserved a blue boat B in 2015. The boat B is reserved again by another sailor in 2016.

$$\rho(temp1(sid1 \leftarrow sid), \pi_{color,sid}(Boats \bowtie Reserves))$$
$$\rho(temp2(sid2 \leftarrow sid), \pi_{color,sid}(Boats \bowtie Reserves))$$
$$\rho(temp3(sid3 \leftarrow sid), \pi_{color,sid}(Boats \bowtie Reserves))$$
$$\pi_{color}(\sigma_{(sid1 \neq sid2)}\sigma_{(sid1 \neq sid3)}\sigma_{(sid3 \neq sid2)}(temp1 \bowtie temp2 \bowtie temp3))$$

Q10: What does the algebra above yield?

colors that have been chosen by at least three different sailors in their reservations

colors that have been chosen by at least three sailors in their reservations

the three most common boat colors among boats

1

This is similar to the self join we did for the 6 degrees of separation in worksheet for SQL where we make sure the paths length is 6 --- here there must be 3 different sailors.