Homework 3 – Answer

10.7

With the unclustered hash index on StudId, we will find exactly the bucket that contains all the transcript records for student with the Id 666666666. Since the index is unclustered, this access method will fetch (in the worst case) as many pages as the number of transcript records for the student. Thus, the selectivity would be this many pages of data.

With the unclustered hash index on Semester, we jump to the bucket for the transcript records in the F1995 semester and then we need to fetch all these records from the disk to check the other conditions. In general, the number of records for a semester would be much higher than the number of records of a student (typically, a student needs to take around 50 courses to graduate whereas this number would be easily in the 1000s for a school). As such, the selectivity of this access path will very likely be in the 1000s.

With the unclustered hash index on Grade, we get into the bucket of the transcript records where the student received the grade A. Under a very reasonable assumption that only 20% of students get A per semester, a school with only 1000 students will have 200 records for one year. Thus, the selectivity of this access path will very likely be in the 100s (if not 1000s).

Thus, in a typical university, the first access path has the best selectivity.

Although we might think that the second access path should have the worst selectivity, it could be the third one as well. This happens when the relation contains also historical data. While the selectivity for the second access path does not change, it can be changed for the third one: for a school of 1000 students, 20 years data yields 4000 (200×20) instead of 1000.

10.8

1. Nested loops: scan r and for each of its 40,000 tuples scan s once. The result is

$$2,000 + 40,000 \times 5,000 = 200,002,000 \text{ pages}$$

2. Block-nested loops: Scan s once per each 400-page block of r, i.e., 5 times. The result therefore is:

$$2,000 + 5,000 \times \lceil 2,000 / (402 - 2) \rceil = 27,000 \text{ pages}$$

3. Index-nested loops: The result depends on whether the index on B in s is clustered or not. For the clustered case, all tuples of s that match a tuple of r are in the same disk block and require 1 page transfer (since we assumed that at most 5 tuples of s

match, they all fit in one page). We also need to search the index once per each tuple of r. Suppose the later takes 1 disk access. Thus, the total is

$$2,000 + (1 + 1 \times 1.2) \times 40,000 = 90,000 \text{ pages}$$

In case of an unclustered index, the matching tuples of s can be in different blocks. As before, assume that s has at most 5 matching tuples per tuple in r. Thus, the cost would be

$$2,000 + (1 + 5 \times 1.2) \times 40,000 = 282,000 \text{ pages}$$

10.13

1. $\sigma_{A>3000}$
   - Sequential scan: Since the index is clustered, the relation is sorted, so to scan the tuples with A < 3000 we need to fetch
     $100,000 \times 3,000 / 1,000,000 = 300$ pages
   - Index on A: Use the index to find the last page page where A < 3000 and then scan the relevant pages. The cost is 4 to search the index plus the cost of the scan, which is the same as above:
     $4 + 300 = 304$ pages

2. $\sigma_{A>3000 \wedge A<3200 \wedge B=5}$
   - Index on A: Use the index to find the first page where A > 3000 then scan the pages until you find the first page where A >3200:
     $4 + 100,000 \times 200 / 1,000,000 = 24$ pages
   - Index on B: On the average there are 1,000,000/100,000 = 10 tuples per any given value of B. Use the hash index on B to find the tuples where B=5. Then we need one page transfer for each tuple in the buffer. In total about
     $1.2 + 10 = 11.2$ pages.

3. $\sigma_{A6\neq22 \wedge B6\neq66}$
   - Sequential scan: Need to scan the entire relation: 100,000 page transfers.
   - Index on A: Not helpful. For every value in the range of 1 to 1,000,000, except 22, use the index to get the page that holds that tuple:
     4(depth of B+ tree) $\times 999,999 = 3,999,996$ pages.
   - Index on B: This index is also not of much use. For each value in the range of 1 to 100,000, except 66, find the bucket that contains pointers to the actual tuples: $1.2 \times 99,999$. For each such tuple (there are about 999,990 of them) fetch the corresponding page. The total cost is thus
     $1.2 \times 99,999 + 999,990 = 1,119,989$ pages