Assignment-9 Query Execution

Solutions CS411 Summer ‘18

---

1 - Calculate number of blocks, B(R)

For a clustered relation R, what is the number of blocks it occupies on disk B(R), if the cost of performing two-way merge sort on R is 120 disk accesses? Assume that B(R) ≤ M^2, where M is the number of blocks that fit in the main memory and please ignore the cost of writing the final sorted output to the disk.

- noFigure

✖ 360

✖ 60

✔ 40

- For a relation R having B(R) blocks, the cost of performing a two-way merge sort is 3B(R). In the first stage, the relation is read into the disks, M blocks at a time. This is done for B(R) blocks, giving a cost of B(R). Next, the blocks are sorted in-memory and written back to the disk. This adds a cost of B(R) again. In the third stage, the different sorted blocks are merged. This is done for (M-1) blocks at a time, leaving a single block of space in memory to write the output. Note that again B(R) blocks are involved in the merge process. Therefore, the total cost of the operation is 3B(R). In the question, we are given 3B(R) = 120, which on simplification gives B(R)=40.

✖ 240

---

2 - Finding records in a clustered table with clustered index

Consider a table Students(netid, major, favorite_restaurant) with 100000 records stored in 1000 blocks on the disk. Assume that there are a total of 50 distinct values for the favorite_restaurant attribute. Calculate the approximate cost of finding the records with favorite_restaurant = ‘cravings’ if the table Students is clustered and there exists a clustered index on the attribute ‘favorite_restaurant’.

- noFigure

✖ 1000 disk blocks

✖ 100,000/50 = 2000 disk blocks

✔ 1000/50 = 20 disk blocks

- Students is a clustered table with a clustered index on the favorite_restaurant attribute meaning that the table is stored such that the records with the same values of favortite_restaurant are adjacent to one another. Since there are 50 distinct values of favorite_restaurant possible across 1000 blocks, we need to do only 20 disk blocks in the average case to find that block containing records with cravings as the favorite restaurant. Once this block is found, we can read it sequentially to find all the student records.

✖ 100,000 disk blocks

---

3 - Cost of block-based nested loop join

Suppose we have two tables Students(netid, major) and Courses(major, number) with major being a common attribute. The table Students occupies 100 blocks on the disk and the table Courses occupies 400 blocks on the disk. We also know that the memory can fit in 50 blocks. We now want to perform a **block-based nested loop** join Students ⋈Courses. In general, block-based nested loop involves the following steps:

1. Bring M blocks of one table to the memory.
2. Perform join with all the blocks in the other table, one block at a time.
3. Repeat Step 1 until no more blocks are left in table in Step 1.

Which table would you use in the Step 1 and what is the minimum cost of the operation?

- noFigure

✖ Students, 1200

✔ Students, 900

- Nested loop join R ⋈ S is performed by bringing M blocks of S into memory and performing join with B(R)blocks of R, one block at a time. The cost of this operation is B(S)+B(R)∗B(S)/M disk blocks. Clearly, this quantity will be smaller if B(S) is smaller. Thus it benefits to use the smaller relation in Step 1. The smaller relation is Students, occupying only 100 blocks compared Courses which spans 400 blocks. The minimum cost of this operation is 100 + 400*100/50 = 900 disk blocks.

✖ Courses, 1200

✖ Courses, 900

✖ Does not matter, 900

---

4 - Cost of hash-based join

What is the cost of performing a hash-based join on relations R and S where R fits in main memory, but S is much larger than main memory and both R and S are clustered?

- noFigure

✔ B(R) + B(S)

- We can first bring R into the main memory costing us B(R) disk accesses. Then, we can bring S into the main memory one block at a time. For every block of si of S with a hash value h since S is clustered, the join costs only 1 access as the relation R is clustered. Therefore, we get B(R) + B(S) *1 = B(R) + B(S).

✖ 2(B(R) + B(S))

✖ 3B(R)

---

5 - Viability of hash-based join

Suppose we have two relations, R(x,y) and S(y,z). We know that R occupies 400 blocks on disk and S occupies 300 blocks. Neither R nor S are sorted on any of their attributes. The memory buffer fits 50 blocks (M=50). Suppose we want to join R and S using a partitioned hash join. Is this possible?

- noFigure

✖ Possible, since 400 > 50.

- See the correct answer for calculation.

✔ Possible, since 300/50 ≤ 50.

✖ Impossible, since 50 < 300.

- See the correct answer for calculation.

✖ Impossible, since 400/50 > 300/50.

- See the correct answer for calculation.

---

6 - Memory requirement of 2-pass multi-way merge sort

Assume a clustered relation R has 30,000 tuples with 100 tuples per block. What should be the minimum number of blocks in main memory that allows the two-pass merge sort operation on R?

- noFigure

✖ None

✖ 30

✖ 17
✔ 18

- We are given T(R)=30000. Each block can hold 100 tuples. Therefore, we have B(R)=300. For two-pass merge sort operation, the requirement is that $M^2 \geq B(R)$. Therefore, the value of M that gives this is 18 since 18*18 = 324 > 300.

---

7 - Physical and logical operators

In the lectures, Prof. Chang mentioned about logical and physical operators in query execution plans. Logical operators refer to what a query does and physical operators are how a query processing engine accomplishes it. Therefore, logical operators include union, projection and grouping whereas physical operators include nested loop join, hash join and sort-merge join. This statement is:

- noFigure

✔ True

- For an operation, logical operators only describe the input and output whereas physical operators describe the mechanism by which the operation is carried out. For example, join is a logical operator whereas hash-based join is a physical operator.

✖ False

---

8 - Calculate M

Consider two relations, R and S, with B(R) = B(S) = 10,000, i.e., both relations occupy 10,000 blocks on disk. What is the minimum value of M, i.e., memory, required to compute the join results of R and S using a block-based nested loop join with no more than 35,000 block disk accesses? Choose the correct inequality to calculate M:

- noFigure

✖ 5 * B(R) * M ≥ 35,000
✖ None
✖ 3 * (B(S) + B(R)) * M ≥ 35,000
✖ 3 * B(R) * M ≥ 35,000
✔ B(R) + B(R) * B(S)/M ≤ 35,000

- The cost of block-based nested loop join of R and S is B(R) + B(R) * B(S)/M which is required to be less than 35,000 as specified in the question.

---

9 - Sort-merge join

How should we sort the data when performing a sort-merge join?

- noFigure

✖ Based on the whole tuple, from the first to the last attribute.

✖ Based on the key of each relation.

✔ Based on the join attribute.

- Once the relations are in sorted order, tuples with the same value on the join attributes are in consecutive order. Therefore, each tuple in the sorted order needs to be read only once, and, as a result, each block is also read only once. This makes sort-merge join efficient costing B(R) + B(S) disk accesses.

---

10 - Choose viable operations with given constraints

Suppose we want to join two clustered relations R and S with B(R)=250 and B(S)=400. Which of the following algorithm is applicable if the memory capacity M is 15?

- noFigure

✖ Sort-merge join

✔ Nested loop join

- Sort and hash-based join requires the condition $M^2 \geq \min(B(R), B(S))$. However, in this case we see that $M^2 = 225 < \min(B(R), B(S))$. Therefore, we can do only nested loop join by bringing M blocks of S into memory and performing join with B(R) blocks of R, one block at a time. This costs B(S) + B(R) * B(S)/M disk accesses.

✖ Hash-based join

✖ None
