Weekly exercises IN3020&4020 (2025)
Week 7: Query Evaluation

1 Review

1. What is the difference between logical query plan and physical query plan?
2. What is our cost for evaluation of an operation (not to be confused with the cost of a logical plan for optimisation)? What are our main assumptions?
3. Assuming you want to retrieve the entire relation, what is the IO cost of using a table scan? Same for the index scan?
4. Outline the main ideas of Two-Phase Multiway Merge Sort (TPMMS) and Hash Partitioning. What is their costs and conditions/requirements?
5. What are the main advantages and disadvantages of one-pass algorithms, the two variants of two-pass algorithms (sort-based, hash-based), and index-based algorithms?

2 Unary operations

1. Assume $Tups(R) = 10,000$, $Blocks(R) = 500$ and $Vals(R.A) = 100$ for a relation R with an attribute A. What is the cost of our algorithm for $\sigma_{A=c}(R)$ if there are no indexes on A? If there exists an index on A (consider all types of indexes we know, but assume that accessing the index is of no cost)?
2. Let $Tups(R) = 5,000,000$ and $Blocks(R) = 100,000$ for a relation R, and there are 1000 buffers (i.e., blocks in the main memory). What is the cost of $\delta(R)$?

3 Blocking Operations

1. For each of the operators generalised bag projection, duplicate elimination, grouping and aggregation, set and bag union, intersection, difference, Cartesian product, and natural join, tell whether the operator is *blocking*, by which we mean that the first output cannot be produced by our evaluation algorithms until all the input has been read (assuming that each the argument relations occupy more than 1 block).

4 One-pass algorithms for binary operations

1. Outline one-pass algorithms for $R \cup^b S$, $R \bowtie_\theta S$, and $R \times S$. Give the cost and space requirements for these algorithms.
2. Can nested-loop approach be applied to theta-join? If so, what will be the cost and space requirements?

5 Sort-based algorithms

1. Let $Blocks(R) = Blocks(S) = 10,000$ for relations R and S, and $MMBuffers = 1000$. What is the cost of our algorithms for set union and two-pass sort-based natural join of R and S?
2. How many main-memory blocks do we need to use for a two-pass sort-based algorithm for relations of 10,000 blocks (per relation) if the operation is duplicate elimination, grouping with aggregation, set union, natural join?
3. Design an algorithm that optimises the two-pass sort-based natural join $R \bowtie S$ that combines the second phase of the sorts with the join itself.
4. (*) For the two-pass-sort-based natural join algorithm, we have a requirement that there should be at most $MMBuffers/2$ blocks with the same join value in each relation. Explain what should be changed in the algorithm if this requirement does not hold and how it influences the cost. Consider two cases: when each join value appears in at most $MMBuffers$ blocks (in both relations together) or not. (Note: you do not need to describe the new algorithms in full, just discuss its essential differences)

6 Hash-based algorithms

1. Come back to slide 41 of the slide set Lecture_13to15 and analise two-pass hybrid hash-based natural join algorithm. Try to understand

why the improvement ensures the gain in the cost. If not sure, look at Section 15.5.6 in Garsia-Molina book (see slide 4 in the slide set for the complete reference).

2. The hybrid-hash-join idea of storing one bucket in main memory can also be applied to other operations. Show how to save the cost of storing and reading one bucket from each relation when implementing a two-pass hash-based algorithm for duplicate elimination, grouping and aggregation, and intersection.

3. (*) Let $Blocks(R) = Blocks(S) = 10,000$ for relations R and S, and $MMBuffers = 1000$. What is the number of disk I/O 's required for the two-pass hybrid hash join of R and S in the case when the number of hash values (i.e., buckets) is 11 (and the tuples of the relations are evenly distributed over the buckets)?

7 Index based algorithms

1. Suppose there is an index on attribute A of relation R and another relation S. Describe how this index could be used to improve the execution of $\delta(R)$ and $R \cap S$. Under what circumstances would the index-based algorithm be more efficient than sort- or hash-based algorithms?

2. Let $Blocks(R) = 10,000$ and $Tuples(R) = 500,000$ for a relation R. Let there be an index on attribute A of R and let $Val(R.A) = k$ for some number k. Give the cost of $\sigma_{A=0}(R)$ as a function of k, when the index is clustering, secondary, or not used at all (where you may neglect disk I/O's needed to access the index itself).