Tutorial 5 — Query Optimization, Planning, Evaluation

Richard Wong
rk2wong@edu.uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

March 4, 2018

Exercise 5-1

1. What are the two metrics we will use to estimate query operation costs?
2. What does each metric represent?
3. How do we use the metrics to arrive at an estimate?

Exercise 5-1 Solution

1. In this course, we care primarily about cost contributed by block transfers and by disk seeks.
2. A block transfer occurs whenever we need to retrieve a block from disk to read in memory, or when we need to write a block back to disk.
   A disk seek occurs when we need to access a disk block non-sequentially.
3. Let block transfers take time $t_{transfer}$ and disk seeks take time t_{seek}.
   Let there be b block transfers and s disk seeks.
   Then we estimate the cost of a query operation with the formula
   $b(t_{transfer}) + s(t_{seek})$.

Exercise 5-2

Suppose we run a query that performs a single-attribute GREATER THAN comparison in its WHERE clause.

e.g.
SELECT * FROM people
WHERE age > 20

How might the following evaluation strategies impact the (worst-case) cost of the operation?

1. Use a primary index if there is one.
2. Use a secondary index if there is one.
3. Use a linear scan.

What if we were performing a LESS THAN comparison?

Exercise 5-2 Solution

Assuming our indices are B+ trees, let h_{index} be the height of the index.
Let B be the number of blocks in the relation.
Let b be the number of blocks containing records that satisfy the condition.
Let n be the number of records that satisfy the condition. ($n \geq b$)

1. Primary index: $h_{index}(t_{seek} + t_{transfer}) + b(t_{transfer})$
   h_{index} seeks and transfers to locate the starting point with the index.
   Since this is a primary index, the last seek in the index brings us straight to the data, and no additional seek is required.
   b transfers to read the sequentially-organized data, following leaf node pointers.
2. Secondary index: $(h_{index} + n)(t_{seek} + t_{transfer})$
   h_{index} seeks and transfers to locate the starting point with the index.
   Since this is a secondary index, we need a seek and a transfer for each block. In the worst case, each subsequent record is located on a different block.
   n seeks and transfers for those blocks.
3. Linear scan: $t_{seek} + B(t_{transfer})$
   1 seek, B transfers to read the whole relation. No index $\rightarrow$ no ordering assumption.

Exercise 5-2 Solution, continued

If we were performing a LESS THAN as opposed to a GREATER THAN comparison, using the primary index will give no benefit, since we only need to seek to the beginning of the relation and perform a linear scan (assuming ascending order of index).

Note that using a secondary index could result in a greater cost than a linear scan, especially if $n >> b$.

Suppose there are b_r blocks in R and b_s blocks in S.
Derive the worst-case and best-case cost estimate for a block nested-loop join, $R \bowtie_\theta S$.

Exercise 5-3 Solution

In the worst case, we can assume that we have enough memory to hold exactly 1 block of R and exactly 1 block of S in main memory.

For each block in R, of which there are b_r, we have to:

1. **seek** to the block in R,
2. **transfer** that block to main memory, then
3. **seek** to the beginning of S.
4. then for each block in S, we need to:
    1. **transfer** that block to main memory, then
    2. do some CPU work to accumulate the R, S record pairs that match the predicate.

So the join cost would be $2b_r(t_{seek}) + b_r(1 + b_s)(t_{transfer})$.

Exercise 5-3 Solution, continued

In the best case, we can assume that we have enough memory to hold all of R and all of S in main memory, so we just need to:

1. **seek** to the beginning of R,
2. **transfer** the b_r blocks of R to main memory,
3. **seek** to the beginning of S,
4. **transfer** the b_s blocks of S to main memory, then
5. do some CPU work to accumulate the result.

So the join cost would be $2(t_{seek}) + (b_r + b_s)(t_{transfer})$.