A

CSE 444: Database Internals.
Section 10: Review.

1. QUERY COST ESTIMATION

Consider the following three relations:

(1) $R(w, x)$: 1000 blocks, 10 tuples per block.
(2) $S(x, y)$: 10000 blocks, 10 tuples per block, $V(S, y) = 1000$.
(3) $U(y, z)$: 10000 blocks, 10 tuples per block, $V(U, z) = 1000$.

Query: $(\sigma_{z=c}(U) \bowtie S) \bowtie R$.
Assume that you have 2000 blocks of memory.
The inner join is a nested loop (block at a time). The other join is a hash join.
Find the number of disk IOs:

(1) Assume no other indices:
Answer: Due to the $U.z$'s selectivity, only 100 tuples, in 10 blocks, are selected from U. Since we lack any indices on U, we scan the whole of U. For **each block** of filtered tuples from U, we scan the whole of S (we lack indices on S as well) and perform the join. The total output of joining U and S is 10000 tuples which in memory. Thus, we read in R and probe into the hash table. Thus, the total number of page IOs is 10000 + 10000 / 1000 * 10000 + 1000 = 111000.
(2) Assume unclustered index on y in S and z in U:
Answer: Due to the $U.z$'s selectivity, we only read to 100 tuples from U. Since we use an unclustered index, this amounts to 100 blocks from U. For **each tuple** t read from U, we need to read in 100 tuples from S that have the same value for attribute y as tuple t has for y. Since we also have an unclustered index on $S.y$, we can retrieve the 100 tuples by reading 100 blocks. The total output of joining U and S is 10000 tuples which in memory. Thus, we read in R and probe into the hash table. 100 + (10000 / 1000) * 10 * 100 + 1000 = 11100.
[Unlike the case without indices where we iterated over the entire relation R for **each block** of filtered tuples from U, with indices, we have to operate on each of U's filtered **tuples individually**. This is because, each of the filtered tuples of U may have a different value for the join attribute (note that the filter works on the attribute z while the join is on the attribute y) and thus, the tuples retrieved from S for a certain tuple in a block of U may be unusable for the other tuples in thatx block.]
(3) Assume clustered index on y in S and z in U:
Answer: Due to the $U.z$'s selectivity, we only read to 100 tuples from U. Since we use a clustered index, this amounts to 10 blocks from U. For **each tuple** t read from U, we need to read in 100 tuples from S that have the same value for attribute y as tuple t has for y. Since we also have a clustered index on $S.y$, we can retrieve the 100 tuples by reading 10 blocks from S. The total output of joining U and S is 10000 tuples which in memory. Thus, we read in R and probe into the hash table. 100 + (10000 / 1000) * 10 * 10 + 1000 = 2010.

2. SELINGER OPTIMIZER

Consider a Selinger style optimizer. Given the following relations:

(1) $R(w, x)$: 1000 blocks, 10 tuples per block.
(2) $S(x, y)$: 10000 blocks, 10 tuples per block.

(3) $U(y,z)$: 10000 blocks, 10 tuples per block.

Compute $R \bowtie S \bowtie U$. For the cost function, use the number of page IOs.
You may assume the following about the optimizer:

(1) No cross-product.
(2) Assume that join selectivity is 0.01%.
(3) Assume only sort-merge join can be used.
(4) No indexes.

Find the optimal plan.

Answer:. We represent the cost and output cardinality (in blocks) of a sub-plan by $(cost, cardinality)$.

Trees of size 1:

R = (1000, 1000). S = (10000, 10000). U = (10000, 10000).

Trees of size 2:

$R \bowtie S$: Cost = Cost of reading R + cost of reading S + cost of sorting them + cost of merging them = 1000 + 10000 + 1000 + 10000 + 1000 + 10000 = 33000. Output size = 100000 = 0.1M = 10K blocks. Thus, $R \bowtie S$ = (33K, 10K). Note that we need not consider the plan $S \bowtie R$ since sort-merge joins are symmetric.

We do not consider the sub-plans $R \bowtie U$ and $U \bowtie R$ since this is a cross product.

$S \bowtie U$: Cost = 30000 (for S) + 30000 (for U). Output size = 1000000 = 1M = 100K blocks. Thus, $S \bowtie U$ = (60K, 100K).

Trees of size 3:

$(R \bowtie S) \bowtie U$: Cost = 33000 (from $R \bowtie S$) + 20000 (sorting and merging of $R \bowtie S$) + 30000 (reading, sorting, and merging U). We need to sort again since the join attribute is not the one sorted in the sub-plan. Thus, the total cost is 83K.

$(S \bowtie U) \bowtie R$: Cost = 60000 + 200000 + 3000 = 263K.

3. MULTI-VERSION CONCURRENCY CONTROL

What happens with concurrency control with multiple version timestamp based scheduler, in each of the following cases?

(1) $st_1; st_2; st_3; st_4; w_1(A); w_2(A); w_3(A); r_2(A); r_4(A)$

Answer:
[A_1 created.] $w_1(A)$
[A_2 created.] $w_2(A)$
[A_3 created.] $w_3(A)$
[Reads A_2.] $r_2(A)$
[Reads A_3.] $r_4(A)$

(2) $st_1; st_2; st_3; st_4; w_1(A); w_3(A); r_4(A); r_2(A)$

Answer:
[A_1 created.] $w_1(A)$
[A_3 created.] $w_3(A)$
[Reads A_3.] $r_4(A)$
[Reads A_1.] $r_2(A)$

(3) $st_1; st_2; st_3; st_4; w_1(A); w_4(A); r_3(A); w_2(A)$

Answer
[A_1 created.] $w_1(A)$
[A_4 created.] $w_4(A)$
[Reads A_1.] $r_3(A)$
[Abort transaction 2.] $w_2(A)$