Homework: Query Optimization (w9)

- Rahul Kumar Nalubandhu

For the four relations in the following table, find the best join order according to the dynamic programming algorithm used in System-R. You should give the dynamic programming table entries for evaluate the join orders. The cost of each join is the number of I/O accesses the database system performs to execute the join. Assume that the database system uses two-pass sort-merge join algorithm to perform the join operation. Each block contains 4 tuples and tuples of all relations have the same size. We are interested only in left-deep join trees. Note that you should use the System-R optimizer formula to compute the size of each join output(2 points).

R(A,B,C)	S(B,C)	W(B,D)	U(A,D)
T(R)=4000	T(S)=3000	T(W)=2000	T(U)=1000
V(R,A) =100 V(R,B) =200 V(R,C) =100	 V(S,B) =100 V(S,C) = 300	 V(W,B) =100 V(W,D) =50	V(U,A) =100 V(U,D) =100

Solution:
So as per the question we have each block contains 4 tuples and tuples of all relations have the same size. So, the number of blocks for each relation can be calculated as :

$B(R) = \frac{T(R)}{4} = \frac{4000}{4} = 1000,$

$B(S) = \frac{T(S)}{4} = \frac{3000}{4} = 750,$

$B(W) = \frac{T(W)}{4} = \frac{2000}{4} = 500,$

$B(U) = \frac{T(U)}{4} = \frac{1000}{4} = 250.$

For Size :
now for calculating the size for relation R & S

$$T(R \bowtie_{\mathrm{B,C}} S) = T(R) * T(S) / \max\big(V(R,B), V(S,B)\big) \max\,(V(R,C), V(S,C))$$

Therefore, size of join R,S = $\frac{4000*3000}{200*300} = 200$.(200/4 = 50 blocks)

For Cost :
The cost for the join of relations R and S can be calculated as

$$5\big(B(R) + B(S)\big) = 5(1000 + 750) = 8750$$

The overall expense of merging three related entities can be determined by adding the cost of a binary join and the expense of merging the result of the binary join with a third entity. If the intermediate entity isn't arranged according to the attribute used for merging, a two-pass merge-sort is employed to sort it.

For relations R, S, W the cost is :
Join S and R based on B and C, so the result will be already sorted on B.
So, there is no need to sort to join with W.

Now For relations R, S, U we can see there is no common attribute to these relations hence in relation U we have A, but relation S and R are not sorted on A. So, it should sort result of the join SR based on A to join it with relation U using the sort-merge algorithm.

The cheapest plan to join all relations is ((S ⋈ R) ⋈ U) ⋈ W. The results of (S ⋈ R) ⋈ U) is not. sorted based on B or D.

hence the **cost is** 5B(W) + 5B((S ⋈ R) ⋈ U)+c((S ⋈ R) ⋈ U)) = 15,250 and the **size is**

$T(W) * T((S \bowtie R) \bowtie U))/\max(V(W,B), V((S \bowtie R) \bowtie U), B)) \max(V(W,D), V((S \bowtie R) \bowtie U), D))$

$= \frac{2000*2000}{100*100} = 400$ (400/4 =100 blocks)

So, by doing this we have lot of plans, but we must consider with the minimum cost so for computing the join of R, S, W, U **we select the plan ((S ⋈ R) ⋈ U) ⋈ W).**

Possible Queries	Size(blocks)	Cost	Plan
R, S	50	8750	S⋈R
R, U	10,000	6250	U⋈R
R, W	10,000	7500	W⋈R
S, U	750,000	5000	U⋈S
S, W	15,000	6250	W⋈S
W, U	5000	3750	U⋈W
R, S, U	500	10,250	(S ⋈ R) ⋈ U
R, S, W	1,000	11,350	(S ⋈ R) ⋈ W
R, W, U	1,000	33,750	(U ⋈ W) ⋈ R
S, W, U	15,000	32,500	(U ⋈ W) ⋈ S
R, S, W, U	100	15,250	((S ⋈ R) ⋈ U) ⋈ W