UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Assignment 5 - Report

Department of Industrial and Enterprise Systems Engineering
Zhenye Na (zna2)
April 30, 2018

1 QUERY EXECUTION (25 PTS)

1. Consider the following relations with no indexes on them:
   - Relation R has 5,000 tuples, 100 tuples per block
   - Relation S has 2,000 tuples, unknown number of tuples per block

   The number of blocks in memory is 10. Say, S is as large as possible (within the limits that main memory can afford) for the two pass sort-merge join (slide 57). That is, if S was larger, the two pass sort-merge join would not work.

   Answer the following questions:

   a) How many tuples per block does S have? (Do not forget to show your calculations.)
   Solutions:
   Number of blocks in R: $B(R) = \frac{5000}{100} = 50$
   Memory requirements for two pass sort-merge join is $B(R) + B(S) \leq M(M-1)$, so we can get $50 + B(S) \leq 10 \times 9$, $B(S) \geq 40 = 40$.
   So, we got $\frac{2000}{40} = 50$ tuples per block.

   b) Using your answer from A, what is the cost of joining R and S using the two pass sort-merge join algorithm (slide 57)?
   Solutions:
   Cost = 3B(R) + 3B(S) = 150 + 120 = 270.

   c) Using your answer from A, what is the cost of joining R and S using the optimized block nested-loop join algorithm?

Solutions:
Cost = $\frac{B(R)B(S)}{M-2} + \min\{B(S), B(R)\} = \frac{50\times40}{8} + 40 = 290$

d) What is the cost of joining R and S using a two pass hash-based join?
Solutions:
Cost = 3B(R) + 3B(S) = 150 + 120 = 270.

e) Based on questions 2, 3, and 4, explain which variant of the algorithm you would choose in terms of I/O cost. If multiple algorithms have the same I/O cost, explain other considerations that may influence your choice.
Solutions:
The I/O costs of two pass sort-merge join algorithm and two pass hash-based join algorithm are the same. However, two pass sort-merge join algorithm is more susceptible to skewed data, so I will choose two pass hash-based join algorithm.

2 Query Optimization (35 pts)

Consider the relations A(x,y,z), B(w,x), and C(u,v,w), with the following properties:

A(x,y)	B(y,z)	C(z,x,u)
T(A) = 2500	T(B) = 1000	T(C) = 6000
V(A, x) = 30	V(B, y) = 250	V(C, z) = 20
V(A, y) = 500	V(B, z) = 100	V(C, x) = 60
		V(C, u) = 40

where, T(R) = number of tuples in relation R and V(R, a) = number of distinct values of attribute a in relation R. Estimate the sizes (measured in number of tuples) of the result of the following expressions:

1. $A \times B$
   Solution:
   2500 × 6000 = 15000000

2. $A \bowtie B$
   Solution:
   $A \bowtie B$ is joining on column y, V(A,y)=500 > V(B,y)=250 so we loop through B.
   For reach tuple in B, it will join $1000 \times \frac{2500}{500} = 5000$ So size = 5000

3. SELECT u FROM C WHERE u=20
   Solution:
   Size will be from 0 to 5941.

4. $\sigma_{x=10 and y=30}(B \bowtie C)$
   Solution:
   We first compute size of $B \bowtie C$:
   $B \bowtie C$ will join on column z. V(B,z) = 100 > V(C,z) = 20 so we loop through C.
   For each tuple in C, it will join $6000 \times \frac{1000}{100} = 60000$
   Then we perform selection Size will be from 0 to (60000-250+1=59751)

3 DYNAMIC PROGRAMMING (40 PTS)

Consider the following relations, where T(R) = number of tuples in relation R and V(R, a) = number of distinct values of attribute a in relation R.

A(x,y)	B(y,z)	C(z,x,u)	D(u,v)
$T(A) = 2500$	$T(B) = 1000$	$T(C) = 6000$	$T(D) = 2000$
$V(A, x) = 30$	$V(B, y) = 250$	$V(C, z) = 20$	$V(D, u) = 100$
$V(A, y) = 200$	$V(B, z) = 100$	$V(C, x) = 60$	$V(D, v) = 40$
		$V(C, u) = 40$	

We want to join all these relations as efficiently as possible. Determine the most efficient way to do the join. Clearly state any assumptions you have made. Show your work by completing the following table (each step in the dynamic programming algorithm should be one row):

Subset	Size	Lowest Cost	Lowest cost plan
...	...	...	...

Subset	**Size**	**Lowest Cost**	**Lowest Cost Plan**
AB	10K	0	BA
AC	250K	0	AC
AD	5M	0	DA
BC	60K	0	BC
BD	2M	0	BD
CD	120K	0	DC
ABC	10K	10K	C(BA)
ABD	20M	10K	D(BA)
ACD	5M	120K	A(DC)
BCD	1.2M	60K	D(BC)
ABCD	200K	130K	(BA)(DC)

Table 3.1: Dynamic Programming Table