CSE 444 – Homework 3
Query Optimization

Name: Yuanfeng Li

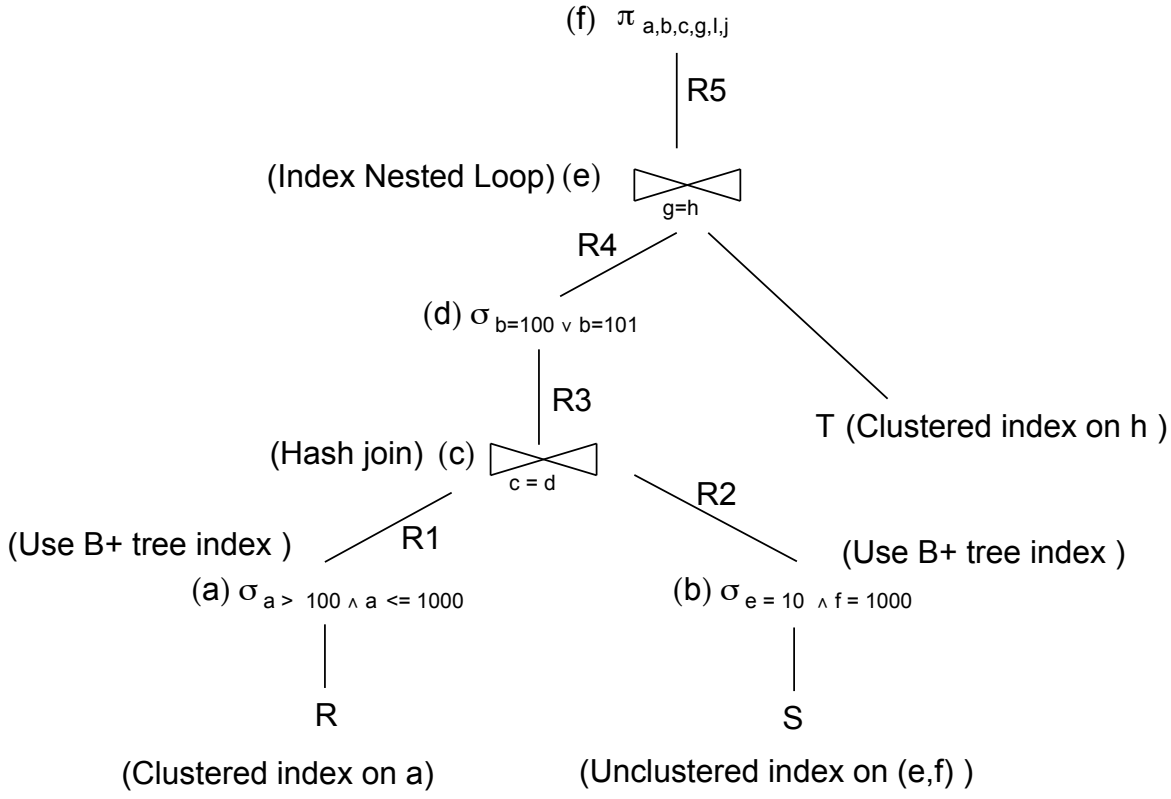
Question	Points	Score
1	20	
2	20	
Total:	40	

1 Query Plan Cost Computation

1. (20 points)

   Consider the following relations and physical query plan:

R(a,b,c)	S(d,e,f,g)	T(h,i,j)
B(R) = 1000	B(S) = 1000	B(T) = 1000
T(R) = 10,000	T(S) = 10,000	T(T) = 10,000
Min(R,a) = 0	V(S,e) = 10	V(T,h) = 100
Max(R,a) = 9000	V(S,f) = 100	
V(R,b) = 100	V(S,d) = 50	
V(R,c) = 20	V(S,g) = 40	



(a) (7 points) Compute the selectivity of the following predicates:

1. $a > 100 \wedge a \leq 1000$
2. $e = 10 \wedge f = 1000$
3. Join predicate: $c = d$
4. $b = 100 \vee b = 101$
5. $g = h$

1. selectivity01 = (1001 − 0)/(9000 − 01) − (101 − 0)/(9000 − 01) = $\frac{900}{9000} \simeq \frac{1}{10}$

2. selectivity02 = $\frac{1}{V(e)} \times \frac{1}{V(f)} = \frac{1}{10} \times \frac{1}{100} = \frac{1}{1000}$

3. selectivity03 = $1 / Max(V(R,c), V(S,d)) = 1/50 = \frac{1}{50}$

4. selectivity04 = $2 / V(R,b) = \frac{2}{100} = \frac{1}{50}$

5. selectivity05 = $1 / Max(V(S,g), V(T,h)) = \frac{1}{100}$

(b) (7 points) Compute the cardinality of all intermediate relations labeled R1 through R5 and the final result, call it R6.

$R_1 = T(R) \times \text{selectivity}_{01} = 10000 \times \frac{1}{10} = 1000$

$R_2 = T(S) \times \text{selectivity}_{02} = 10000 \times \frac{1}{1000} = 10$

$R_3 = R_1 \times R_2 \times \text{selectivity}_{03} = 10 \times 1000 \times \frac{1}{50} = 200$

$R_4 = R_3 \times \text{selectivity}_{04} = 200 \times \frac{1}{50} = 4$

$R_5 = R_4 \times \text{selectivity}_{05} \times T(T) = 4 \times 10000 \times \frac{1}{100} = 400$

$R_6 = 400$

(c) (6 points) Compute the cost of this query plan in terms of number of pages read from disk or written to disk. Assume that all index pages are in memory at any time. Also assume that the hash table for the hash join will fit in memory.

1) Read 1 from Relation R: Since it is clustered index on a, so the I/O cost should be $\frac{1}{10} \times B(R) = \underline{100}$ for R_1

2) Read 2 from Relation S: due to unclustered index on e and f, so the I/O cost should be T(S) x selectivity of (b)

$T(S) \times \frac{1}{1000} = \underline{10}$ for R_2

3) Read 3, 4: R_3, R_4 are happening in the main memory, and we assume that hash tables will fit in the memory, so we need no cost for R_3 and R_4.

4) R5: due to clustered index on h, and index Nested Loop Join.

cost = R4 x B(T) x selectivity of (s = h)

cost = $4 \times 1000 \times \frac{1}{100} = 40$

In Summary: Total cost will be $100 + 10 + 40 = 150$ I/Os

2 Query Optimization

2. (20 points)

Consider the following three relations:

R(a,b,c)	S(d,e)	W(f,g,h)
B(R) = 100	B(S) = 1000	B(W) = 10
T(R) = 1,000	T(S) = 10,000	T(W) = 100

Consider the following SQL Query:

```
SELECT *
FROM R, S, W
WHERE R.a = S.d
AND   R.c = W.h
```

(a) (10 points) Assume that all relations are stored in heap files, there are no indexes, only page-at-a-time nested-loop joins can be used, and the selectivity of each join predicate is 0.1%.

Show the query plan selected by a Selinger-style, bottom-up, dynamic programming optimizer. Use the number of disk IO operations as the cost function.

Since only page-at-a-time nested-loop joins are considered, the output of each join is written back to disk before being passed to the next join. However, for this problem **ignore the cost of writes back to disk.** The costs in your dynamic programming table should only include the cost to read.

Hints:

- Remember that a Selinger-style optimizer will try to avoid Cartesian products in the plan. So do NOT consider such plans.
- The Selinger optimizer will only consider left-deep plans.

Draw the selected plan and show how it is derived. You can use the following table to help you but do NOT worry about computing the exact cost and size values if you don't need exact values to prune plans. In the table, P/K indicates the choice to either prune or keep the subplan. *Hint:* When joining tuples, keep in mind that the tuples get bigger.

Subquery	Cost	Size of output	Plan	P/K
R	100 page IOs	1K records on 100 pages	Sequential scan of R	K
S	1K page IOs	10K records on 1K pages	Sequential scan of S	K
W	10 page IOs	100 records on 10 pages	Sequential scan of W	K
RS	...	10K records on 2K pages	...	
SR	...		...	
...				

More space for your answer:

(b) (5 points) Consider the following small modification to the above query and consider that we add an unclustered B+ tree index on S.d as well as a clustered B+ tree index on R.b. Without re-computing the new best plan, explain how these changes affect the optimization process for this query.

```
SELECT *
FROM R, S, W
WHERE R.a = S.d
AND   R.c = W.h
AND   R.b > 100
```

Since we have unclustered B+ tree index on S.d and clustered B+ tree index on R.b now.

1) The cost of Reading Page for R will be decreased dramatically, because we can use clustered B+ tree index to extract all the tuples for R.b > 100.

2) The unclustered index on S.d might make things slower. It depends on which joining method we are using according to B(S) and T(R):

if T(R) > B(S), the unclustered index might make the cost worse.

if T(R) < B(S), the cost will be better if we chose an index based nested loop join.

(c) (5 points) What is an interesting order? Give one or two examples and explain how they can be useful in getting a better physical query plan.

1) I think Interesting order is a certain ordering of a partial query or process that can reduce the overall I/O cost of the whole plan. The interesting point would be that it might not be the optimal for local cost, but it will lead to a lower cost eventually.

For example:

assume there is a clustered index on W.f, and we have the following query.

```
SELECT  W.f = a , R.a = b
FROM    R , W , S
WHERE   R.a = S.d AND W.f = S.d
```

Since W.c has a clustered index, we might choose to use a merge join for W.f = S.d 's join, then the later join (R.a = S.d) can take the advantage of the sorted result of merge join.

2) 2nd Example: For the original query plan in Q2.a, if we add an group by operater, then it might reduce the total I/O cost by retaining a sorted result for the group by aggregate field.

Subquery	Cost	Size of output	plan	P/K
R	100 page IOs	1K records on 100 pages	Sequential scan of R	K
S	1000 page IOs	10K records on 1K pages	Sequential scan of S	K
W	10 page IOs	100 records on 10 pages	Sequential scan of W	K
RS	B(R)+ B(R)B(S) = 100 + 100 * 1000 = 100,100 page IOs	10K records on 2K pages	Page at a time nested loop join on R ⋈ S on R.a = S.d	K
SR	B(S) + B(S)B(R) = 1000 + 1000 * 100 = 101,000 page IOs	10K records on 2K pages	Page at a time nested loop join on S ⋈ R on S.d = R.a	P
SW			Cartesian Product	P
WS			Cartesian Product	P
RW	B(R)+ B(R)B(W) = 100 + 100 * 10 = 1100 page IOs	100 records on 20 pages	Page at a time nested loop join on R ⋈ W on R.c = W.h	p
WR	B(W)+ B(W)B(R) = 10 + 10 * 100 = 1010 page IOs	100 records on 20 pages	Page at a time nested loop join on W ⋈ R on W.h = R.c	K
(RS)W	Cost(RS) * B(RS)B(W) = 100,100 + 2000 * 10 = 120,100 page IOs	1k records on 300 pages	Page at a time nested loop join (R ⋈ S)W	P
(WR)S	Cost(WR) * B(WR)B(S) = 1010 + 20 * 1000 = 21,010 page IOs	1k records on 300 pages	Page at a time nested loop join (W ⋈ R)S	K

My Plan: