Problem 1: [36 points, 6 each]

Given the following three relations $R1(a, b)$, $R2(b, c)$, and $R3(c, d)$ and associated statistics shown below in the meta data table. Estimate the number of tuples in the result relation for the different queries listed below, namely, T(Q).
$T(R1) = 400$; $V(R1, a) = 50$; $V(R1, b) = 50$
$T(R2) = 500$; $V(R2, b) = 40$; $V(R2, c) = 100$
$T(R3) = 1000$; $V(R3, c) = 50$; $V(R3, d) = 100$

If there are any additional assumptions you need to make to answer any of the questions below, please explicitly state them.

1. Q = $\sigma_{(a=10)}(R1)$.
2. Q = $\sigma_{(a>=10)}(R1)$ (Assume that the range of R1.a is [1, 50].
3. Q = $\sigma_{(a>=10 \text{ AND } b=20)}(R1)$. Again assume the range of R1.a is [1, 50].
4. Q = $R1 \bowtie R2$, where $\bowtie$ represents natural join.
5. Q = $(R1 \bowtie R2) \bowtie R3$.
6. Q = $((\sigma_{(a>=10)}(R1)) \bowtie R2) \bowtie R3$

(1) T(R1)/V(R1,a) = 400/50 = 8

(2) The fraction of range over complete domain is (50 – 10)/50 = 40/50
The result size is T(R1) * 4/5 = 320

(3) We already know the result size of (a>=10) is 320. And V(R1,b) = 50. Thus the result size is 320/50 6 or 7

(4) T(R1)T(R2) / max{V(R1,b), V(R2,b)} = 400*500/50 = 4000

(5) We already know the intermediate result of R1joins R2, and then we have
4000 * T(R3) / max{V(R2, c), V(R3, c)} = 4000*1000/100 = 40000

(6) We still use the intermediate results from the above calculation. So we have
320 * T(R2) / V(R1,b) = 320*500/50 = 3200
3200 *T(R3) / V(R2,c) = 3200 * 1000/ 100 = 32000

Problem 2
You are given the following information about the Executives relationship in a database:

Executives has attributes *ename, title, dname,* and *address*; all are string fields of the same
length. The *ename* attribute is a candidate key. The relation contains 10,000 pages. There are 10 buffer pages in memory available for querying.

Problem 2.1: [30 points, 10 each]
Consider the following query:
SELECT E.title, E.ename FROM Executives E WHERE E.title='CFO'
Assume that only 10% of Executives tuples meet the selection condition.

For each index described below, describe the best query plan and show you calculations for the cost (in I/Os) of it. The best query plan for any given sub-problem *might* not use the index, but it cannot use any other indexes of other sub-problems. Assume that the B+ tree has three levels, with the first level (the root) already in memory, not counting towards the 10 pages that are available for querying.

a) Clustered B+ tree index on E.title
b) Unclustered B+ tree index E.title
c) Clustered B+ tree index on (E.ename, E.title)

Solution: **[3 points for describing your query plan, 7 points for the cost]**

a) The best plan, a B+ tree search, would involve using the B+ tree to find the first title index such that title="CFO", cost= 2. Then, due to the clustering of the index, the relation pages can be scanned from that index' s reference,

   cost = 10000 * 10% + 2500 * 10% (Scanning the index)
   = 1000 + 250 + 2 = 1252 (total cost) .

b) An unclustered index would preclude the low cost of the previous plan and necessitate the choice of a simple filescan, cost= 10000, as the best.
c) Although the order of the B+ index key makes the tree much less useful, the leaves can still be scanned in an index-only scan, and the increased number of tuples per page lowers the I/ 0 cost. Cost= 10000 * 0.5 = 5000.

Problem 2.2: [34 points, 17 each]
Consider the following query:
SELECT E.ename FROM Executives E WHERE E.title='CFO' AND E.dname='Toy';
Assume that only 10% of Executives tuples meet the condition E.title = CFO, only 10% meet E.dname =Toy, and that only 5% meet both conditions.

For each index described below, describe the best query plan and show you calculations for the cost (in I/Os) of it. The best query plan for any given sub-problem *might* not use the index, but it cannot use any other indexes of other sub-problems. Assume that the B+ tree has three levels, with the first level (the root) already in memory, not counting towards the 10 pages that are available for querying.

a) Clustered B+ tree index on (E.title, E.ename)
b) Clustered B+ tree index on (E.ename, E.title, E.dname)

***Solution*: [7 points for describing your query plan, 10 points for the cost]**

(a) Although this index does contain the output field, the dname still must be retrieved from the relational data pages, for a cost of 2 (lookup) + 10000 * 10% + 5000 * 10% = 1502.

(b) Finally, in this case, the prefix cannot be matched with the equality information in the WHERE clause, and thus a scan would be the superior method of retrieval. However, as the clustered B+ tree's index contains all the indexes needed for the query and has a smaller tuple, scanning the leaves of the B+ tree is the best plan, costing 10000 * 0.75 = 7500 I/Os