Question1

A = 10,000 Disk pages. B = 1,000 pages. R = 502 buffer pages. No index, simple heap

For each strategy, provide the formulae you use to calculate your cost estimates.

a) Page-oriented Nested Loops Join. Consider A as the outer relation.

Formula: $Cost = NPages(R) + Npages(R) * Npages(S)$

Since A is outer relation, R = 10,000, S = 1000

So $Cost = 10{,}000 + 10{,}000 * 1{,}000 = 100{,}010{,}000$ I/O's

b) Block-oriented Nested Loops Join. Consider A as the outer relation

Formula = $Cost = Npages(R) + ceil\left(\frac{Npages(R)}{B-2}\right) * Npages(S)$

Since A is outer relation, R = 10,000, S = 1000, Block size B = 502

So $Cost = 10{,}000 + \left(\frac{10{,}000}{502-2}\right) * 1{,}000 = 10{,}000 + 20{,}000 = 30{,}000\ I/O's$

c) Sort-Merge Join

Formula:

$$\mathbf{Cost_{SMJ} = NPages(R) + NPages(R) +}$$

$$\mathbf{2* NPages(R)* ceil(1+ \log_{B-1} ceil(NPages(R)/B)) +}$$

$$\mathbf{2* NPages(S)* ceil(1+ \log_{B-1} ceil(NPages(S)/B))}$$

R = 10,000; S = 1,000; B = 52;

So, $Cost = 10{,}000 + 1000 + 2 * 10{,}000 * ceil(1 + \log_{501} 20)$

$+2 * 1{,}000 * ceil(+1 \log_{501} 3)$ = 11000+2*10,000*2+ 2*1,000*2 = 55000 I/O's

d) Hash Join

Formula: $Cost = Npages(R) + Npages(S) + 2 * (Npages(R) + Npages(S))$

R = 10,000; S = 1,000;

So, $Cost = 10{,}000 + 1{,}000 + 2 * (10{,}000 + 1{,}000) = 33{,}000\ I/O's$

e) What would the lowest possible I/O cost be for joining A and B using any join algorithm and how much buffer space would be needed to achieve this cost. Explain briefly.

The lowest possible I/O is reached is when using hashing joint, one table can sits entirely in the memory. Therefore when hashing we can just read both table only once during the join process.

We also need to use 2 buffer pages to write in and out of the result.

The optimal case would be when buffer size = 1002 page. So this is the case that S can be fits in memory, therefore only hashing will happen.

So, $Optimal\ Cost = 10{,}000 + 1{,}000 = 11000$ I/O

Question 2

Consider a relation with the following schema:

Executives (id: integer, name:string, title:string, level: integer)

The Executives relation consists of 100,000 tuples stored in disk pages. The relation is stored as simple heap file and each page stores 100 tuples. There are 10 distinct titles in the Executives hierarchy and 20 distinct levels ranging from 0-20.

Suppose that the following SQL query is executed frequently using the given relation:

SELECT E.ename

FROM Executives

WHERE E.title = "CEO" and E.level > 15;

Your job is to analyze the query plans given below and estimate the cost of the best plan utilizing the information given about different indexes in each part.

Since there are 100,000 tuples and 100 tuples/page, we have 1,000 pages.

a) Compute the estimated result size and the reduction factor (selectivity) of this query

For E.title:

Formula: $RF = \frac{1}{Nkeys(Col)}$

So, $RF = \frac{1}{10} = 0.1$

For E.level > 15:

Formula: $RF = \frac{High(Col) - value}{High(Col) - Low(Col)}$

So, $RF = \frac{20-15}{20-0} = \frac{1}{4}$

Therefore, $RF_{query} = 0.25 * 0.1 = 0.025, Result\ Size = 0.025 * 1{,}000 = 25\ pages$

b) Compute the estimated cost of the best plan assuming that a clustered B+ tree index on (title, level) is (the only index) available. Suppose there are 200 index pages, and the index uses Alternative 2. Discuss and calculate alternative plans

Formula: $Cost = \big(Npages(I) + Npages(R)\big) * \prod RF_i$

So, in this case, we have $Npages(I) = 200, Npages(R) = 1{,}000, RF_1 = 0.1, RF_2 = 0.25$

The Cost is $(200 + 1{,}000) * 0.1 * 0.25 = 30\ I/O's$

Alternative plan is heap scan of R, with cost of NPages(R) = 1000 I/O's

So this method is less than the alternative plan.

c) Compute the estimated cost of the best plan assuming that an unclustered B+ tree index on (level) is (the only index) available. Suppose there are 200 index pages, and the index uses Alternative 2. Discuss and calculate alternative plans.'

Formula: $Cost = \big(Npages(I) + NTuples(R)\big) * \prod RF_i$

So, in this case we have $Npages(I) = 200, NTuples(R) = 100{,}000, RF_{level} = 0.25$

The Cost is $(200 + 100{,}000) * 0.25 = 25{,}050$ I/O's

Alternative plan is heap scan of R, with cost of NPages(R) = 1000 I/O.

So this method is cost more than the alternative plan.

Question2 Continue:

d) Compute the estimated cost of the best plan assuming that an unclustered Hash index on (title) is (the only index) available. The index uses Alternative 2. Discuss and calculate alternative plans

Formula: $Cost = (NTuples(R)) * \prod RF_i * 2.2$

So, in this case, we have $NTuples(R) = 100{,}000, RF_1 = 0.1, Hash\ const = 2.2$

The cost is $100{,}000 * 0.1 * 2.2 = 22{,}000$ I/O's

Alternative plan is heap scan of R, with cost of NPages(R) = 1000 I/O.

So this method is cost more than the alternative plan.

e) Compute the estimated cost of the best plan assuming that an unclustered Hash index on (level) is (the only index) available. The index uses Alternative 2. Discuss and calculate alternative plans.

Formula $Cost = \big(Ntuples(R)\big) * \prod RF_i * 2.2$

Since hashing cannot deal with range value, we have to calculate cost for finding each specific level, then add them up.

In this case, $Ntuples(R) = 100{,}000, RF_{Single\ level} = \frac{1}{20} = 0.05$

So, for cost of search for one level, we have $Cost = 100{,}000 * 0.05 * 2.2 = 11{,}000$

So there are 4 levels that is >15, we need to sum up the cost:

So $Total\ Cost = 5 * 11{,}000 = 55{,}000$ I/O's

Therefore, the total cost is 55,000 I/O's

Alternative plan is heap scan of R, with cost of NPages(R) = 1000 I/O.

So this method is cost more than the alternative plan.

This is because hashing index perfomrs poorly on the selection of range value.

Question 3

Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances (organized on a per department basis)

Emp(eid: integer, did: integer, sal: integer, hobby: char(20))
Dept(did: integer, dname: char(20), floor: integer, phone: char(10))
Finance(did: integer, budget: real, sales: real, expenses: real)

Consider the following query:

```
SELECT D.dname, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.did=D.did AND D.did=F.did
AND E.sal ≥ 59000 AND E.hobby = 'yodeling';
```

The system's statistics indicate that employee salaries range from 10,000 to 60,000, and employees enjoy 200 different hobbies. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial record in the Finance relation) in the database. Each relation fits 100 tuples in a page. Suppose there exists a clustered B+ tree index on (Emp.did) of size 50 pages.

a) Compute the estimated result size and the reduction factors (selectivity) of this query

For E.hobby:

Formula: $RF = \frac{1}{Nkeys(Col)}$

So, $RF = \frac{1}{200} = 0.005$

For E.sal > 59000:

Formula: $RF = \frac{High(Col) - value}{High(Col) - Low(Col)}$

So, $RF = \frac{60{,}000 - 59{,}000}{60{,}000 - 10{,}000} = \frac{1}{50} = 0.02$

For E.did = D.did and D.dif = F.did:

Formula: RF = 1/(max(Nkeys(Col_A), Nkeys(Col_B))

So, RF = 1/5000

Therefore, $RF_{query} = 0.02 * 0.005 * \frac{1}{5000} * \frac{1}{5000} = \frac{1}{2.5e+11}$

Result size = $\prod Ntuples(R)$ * $\prod RF$ = 50,000*5000*5000*$\frac{1}{2.5e+11} = 5\ tuples$

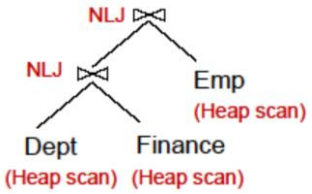
b) Compute the cost of the plans shown below. Assume that sorting of any relation (if required) can be done in 2 passes: 1 st pass to produce sorted runs and 2 nd pass to merge runs. Similarly hash join can be done in 2 passes: 1st pass to produce partitions, 2 nd pass to join corresponding partitions. NLJ is a Page-oriented Nested Loops Join. Assume that did is the candidate key, and that 100 tuples of a resulting join between Emp and Dept fit in a page. Similarly, 100 tuples of a resulting join between Finance and Dept fit in a page.

Star Next Page, With Diagram on it

Question3 b) continue

We have $Npages(Dept) = 50, Npages(Finance) = 50, Npages(Emp) = 500$

1)



Cost of Scanning Dept = $\frac{5000}{100} = 50\ I/O's$, as it is Page Oriented

Cost to Join Dept with Finance using NLJ:

$Npages(Dept) + Npages(Dept) * Npages(Finance) = 50 * 50 + 50 =$ 2550 I/O's

Result Size $\frac{1}{5000} * 5000 * 5000 = 5000\ tuples = \frac{5000}{100} pages = 50\ pages$

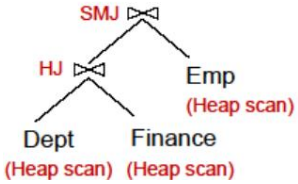
Cost to Join (Dept Join Finance) with Emp:

$Npages(result) * Npages(Emp) = 50 * 500 =$ 25000 I/O's

As we already have result from Dept and Finance in memory.

So, total cost is $2550 + 25000 = 27550\ I/O's$

2)



Cost of Hash Join Dept and Finance:

Formula: $Cost = NPages(R) + NPages(S) + 2 * (Npages(R) + Npages(S))$

In this case $Npages(R(DEPT)) = 50, Npages(S(FINANCE)) = 50$

So, $Cost = 3 * (50 + 50) = 300$ I/O's

Result size is 50 pages

Cost of Join this table with Emp:

Formula:

$Cost_{smj} = Npags(R) + Npages(S) + 2 * Npages(R) * \#pass + 2(Npages(S) * \#pass$

In this case, $Npages(R(DEPT\ HJ\ Finance)) = 50, Npages(S(EMP)) = 500, \#pass = 2$

Cost of Sorting the result Hashing table: 2*50*2 = 200 I/O's

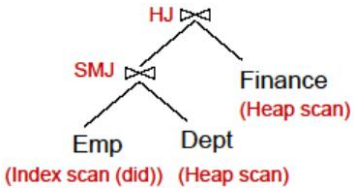
Cost of Sorting the Emp: 2*500*2 = 2000 I/O's

Cost of scanning Emp= 500 I/O's

So, cost of this SMJ is 200 + 500 + 2000 = 2700 I/O's, as we already scanned previous result.

So, Total Cost is $300 + 2700 = 3000\ I/O's$

3)



In this case, we have:

$$Npages(Emp) = 500. Npages(Dept) = 50, Npages(Finance) = 50$$

Cost of Joining Emp with Dept:

Cost of scanning Emp = 500 + 50 = 550 I/O's, as Emp is indexed, no need to sort

Cost of Sorting Dept = 50 * 2 * 2 = 200, as #pass = 2

Cost of Scanning Dept = 50 I/O's

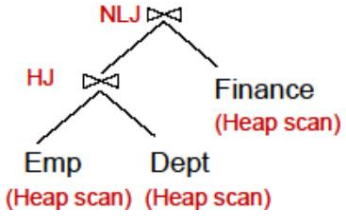
Total cost of SMJ between Emp and Dept = 550 + 50 + 200 =800 I/O's

Result pages size of Emp Joing Dept: = 500, Npages(Finance) = 50

Cost to Hash Join with Finance: 2*500+3*50 = 1150 I/O's, as previous result table is already in memory

So, the total cost is: 800 + 1150 = 1950 I/O's

4)



This case, we have $Npages(Emp) = 500, Npages(Debt) = 50, Npages(Finance) = 50$

Cost of Hash Join Emp and Dept:

3*(500 + 50) = 1650.

Result Page Size = 500 pages.

Cost of NLJ Between result page and Finance:

500*50 = 25000, as previous Result table has already read

Total Cost of operation is:

1650 + 25000 = 26650 I/O's