**Question 1 (5 marks)**

Consider two relations A and B. A is of size 10,000 disk pages, and B is of size 1,000 pages. Consider the following SQL statement:

SELECT \*

FROM A, B

WHERE A.a = B.a;

We wish to evaluate an equijoin between A and B, with an equality condition A.a = B.a . There are 502 buffer pages available for this operation. Both relations are stored as simple heap files. Neither relation has any indexes built on it. Consider alternative join strategies described below and calculate the cost of each alternative. Evaluate the algorithms using the number of disk I/O's as the cost. For each strategy, provide the formulae you use to calculate your cost estimates.

1. Page-oriented Nested Loops Join. Consider A as the outer relation. (1 mark)

Total Cost = (# of pages in outer) + (# of pages in outer) \* (# of pages in inner)

= 10,000 + 10,000 \* 1,000

= 10,010,000 I/Os

1. Block-oriented Nested Loops Join. Consider A as the outer relation. (1 mark)

# of blocks = ceil (# of pages outer / blocksize)

=ceil (10,000 / (502 - 2))

= 20

Total cost = (# of pages in outer) + (# of blocks) \* (# of pages in inner)

= 10,000 + 20 \* 1,000

= 30,000 I/Os

1. Sort-Merge Join (1 mark)

Cost of sorting A = # of pages read (to sort) + # of pages written (after sorting) + # of pages read (to merge) + # of pages written (after merging) = 10,000 + 10,000 + 10,000 + 10,000 = 40,000 I/Os

Cost of sorting B = # of pages read (to sort) + # of pages written (after sorting) + # of pages read (to merge) + # of pages written (after merging) = 1,000 + 1,000 + 1,000 + 1,000 = 4,000 I/Os

Total Cost = Cost of sorting A + Cost of sorting B + # of pages read of A + # of pages read of B = 40,000 + 4,000 + 10,000 + 1,000

= 55,000 I/Os

1. Hash Join (1 mark)

Total Cost = 3(M+N)

= 3 \* (10,000 + 1,000) = 33,000 I/Os

1. What would the lowest possible I/O cost be for joining A and Busing any join algorithm and how much buffer space would be needed to achieve this cost? Explain briefly. (1 mark)

Storing the entire smaller relation (B) in the memory and read each page in the larger relation (A). We could search each tuple in the smaller relation (B) which is matching in the larger relation (A). And the buffer space will hold entire small relation and extra 2 pages, one for reading in the larger relation and one for serving as an output buffer.

Total Cost = M + N

= 10,000 + 1,000 = 11,000 I/Os

Buffer space for this cost = N + 1 + 1

= 1,000 + 1 + 1

= 1,002

**Question 2 (5 marks)**

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 planutilizing the information given about different indexes in each part.

1. Compute the estimated result size and the reduction factor (selectivity) of this query (1 mark)

Reduction Factor(title) = 1/NKeys(Col) = 1/10

Reduction Factor(level) = (High(Col) –value)/(High(Col) –Low(Col))

= (20-15) / (20-0)

= 0.25

Result size = Max # tuples \* product of all RFs’

= 100,000 \* 1/10 \* 0.25

= 2,500

1. 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. (1 mark)

Reduction Factor(title) = 1/NKeys(Col) = 1/10

Reduction Factor(level) = (High(Col) –value)/(High(Col) –Low(Col))

= (20-15) / (20-0)

= 0.25

Cost = (NPages(I)+NPages(R)) \* product of RF’s of matching selects

= (200 + (100,000 / 100)) \* 1/10 \* 0.25

= 30 I/Os

Cost to Heap scan = NPages(R) = 1000, which is greater than Alternative 2. So Alternative 2. is better.

1. 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. (1 mark)

Reduction Factor(level) = (High(Col) –value)/(High(Col) –Low(Col))

= (20-15) / (20-0) = 0.25

Cost = (NPages(I)+NTuples(R)) \* product of RF’s of matching selects

= (200 + 100,000) \* 0.25

= 25,050 I/Os

Cost to Heap scan = NPages(R) = 1000, which is less than Alternative 2. So Heap scan is better.

1. 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. (1 mark)

Reduction Factor(title) = 1/NKeys(Col) = 1/10

Cost = (NTuples(R)) \* product of RF’s of matching selects \* 2.2

= 100,000 \* 1/10 \* 2.2

= 22,000 I/Os

Cost to Heap scan = NPages(R) = 1000, which is less than Alternative 2. So, Heap scan is better

1. 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. (1 mark)

Reduction Factor(level) = (High(Col) –value)/(High(Col) –Low(Col))

= (20-15) / (20-0)

= 0.25

Cost = (NTuples(R)) \* product of RF’s of matching selects \* 2.2

= 100,000 \* 0.25 \* 2.2

= 55,000 I/Os

Cost to Heap scan = NPages(R) = 1000, which is less than Alternative 2. So Heap scan is better

**Question 3 (10 marks)**

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 aclustered B+tree index on (Emp.did) of size 50 pages.

1. Compute the estimated result size and the reduction factors(selectivity) of this query (2marks)

Reduction Factor (E.sal >= 59000) = High(Col) –value)/(High(Col) –Low(Col))

= (60,000 – 59,000) / (60,000 – 10,000)

= 1 / 50

Reduction Factor(E.hobby = ‘yodeling’) = 1/NKeys(Col)

= 1/200

Reduction Factor (E.did = D. did) = 1/ (Max (NKeys(Col\_A), NKeys(Col\_B)))

= 1/ (MAX (5,000, 5,000))

= 1/5,000

Reduction Factor (D.did = F.did) = 1/ (Max (NKeys(Col\_A), NKeys(Col\_B)))

= 1/ (MAX (5,000, 5,000))

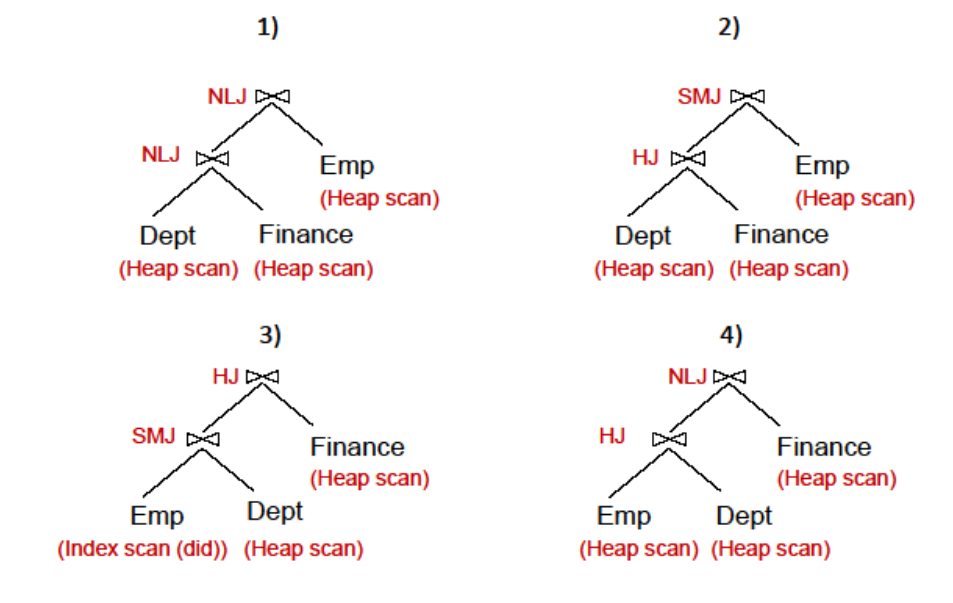
= 1/5,000

Result size = the products of NTuples(Ri) \* the products of RFi

= 50,000 \* 5,000 \* 5,000 \* 1/50 \* 1/200 \* 1/ 5,000 \* 1/5,000

= 5

1. Compute the cost of the plans shown below. Assume that sorting of any relation (if required) can be done in 2 passes: 1st pass to produce sorted runs and 2nd pass to merge runs. Similarly hash join can be done in 2 passes: 1stpass to produce partitions, 2ndpass 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. (8 marks, 2 marks per plan)



1. Nested Loops Join:

Number of Tuples for Dept join Finance = 1/ Keys(I) \* NTuples(R) \* NTuples (S)

= 1/5,000 \* 5,000 \* 5,000

= 5,000

Number of pages for Dept join Finance = 5,000 / 100 = 50

Cost of scanning Dept = 5000 / 100 = 50

Cost to join Dept with Finance = 50 \* (5,000 / 100) = 2,500

Nested Loops Join:

Cost to join Emp = 5,0 \* (50,000 / 100) = 25,000

Total cost = 50 + 2,500 + 25,000 = 27,550 I/Os

1. Cost of scanning Dept = 5000 / 100 = 50

Hash Join:

Cost to join Dept with Finance = NPages(R) + NPages(S) + 2\*(NPages(R) + NPages(S)) = 2\*(5,000/100) + 3\*(5,000/100)

= 250

Number of Tuples for Dept join Finance = 1/ Keys(I) \* NTuples(R) \* NTuples (S)

= 1/5,000 \* 5,000 \* 5,000

= 5,000

Number of pages for Dept join Finance = 5,000 / 100 = 50

Sorted Merge Join

Cost to sort Emp = NPages(R) + NPages(S) + 2\*(NPages(R) + NPages(S))

=(50,000/100) + (50,000/100)+ (50,000/100)+ (50,000/100)

= 2,000

Cost to sort joined Dep with Finance = 50 + 50+50 +50

= 200

Cost to join with sorted Emp = 50 + (50,000/100) = 550

Total cost to join with Emp = 2,000 + 200 + 550 = 2,750

Total cost = 50 + 250 + 2,750 = 3,050 I/Os

1. Sorted Merge Join:

Cost to index scan Emp = (NPages(I)+NPages(R)) \* products of RFi

= (50 + (5,000/100)) \*1/50 \* 1/200 \* 1/ 5,000 \* 1/5,000

= 1/2,500,000,000

Cost to sort Dept = (5,000/100)+(5,000/100)+(5,000/100)+(5,000/100)

= 200

Cost to join with Dept = (50,000/100) + (5,000/100)

= 550

Total cost to join with Dept = 1/2,500,000,000 + 200 + 550 = 750

Number of tuples for Emp join Dept = 1/ Keys(I) \* NTuples(R) \* NTuples (S)

= 1/5,000 \* 50,000 \* 5,000

= 50,000

Number of pages for Emp join Dept = 50,000 / 100 = 500

Hash join:

Cost = 2\*500 + 3\*(5,000/100) = 1,150

Total cost = 750 + 1,150 = 1,900 I/Os

1. Cost to scan Emp = 50,000 / 100 = 500

Hash join:

Cost to join with Dept = 2\*500 + 3\*(5,000/100) = 1,150

Number of tuples for Emp join Dept = 1/ Keys(I) \* NTuples(R) \* NTuples (S)

= 1/5,000 \* 50,000 \* 5,000

= 50,000

Number of pages for Emp join Dept = 50,000 / 100 = 500

Nested Loop Join:

Cost to join with Finance = 500 \* (5,000/100) = 25,000

Total cost = 500 + 1,150 + 25,000 = 26,650 I/Os