Takuhiro Kikuchi (900550) Assignment 3

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.

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

Formulae:
Total Cost =(#of pages in outer) + (#of pages in outer) * (# of pages in inner)
(#of pages in outer) = 10,000
(# of pages in inner) = 10,000
Total Cost = 10,000 + (10,000) * (10,000) = 100,010,000

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

Formula:
# of blocks = (# of pages in outer) / (buffer pages - 2)
Total cost = (# of pages in outer) + (# of blocks) * (# of pages in inner)
# blocks = 10,000 / (502 - 2) = 20
Total Cost = 10,000 + 20 * (1,000) = 30,000

. c) Sort-Merge Join **(1 mark)**

Formula:
Cost of sorting A = # of pages read (to sort)+ # of pages written (after sorting) + # of pages read (to merge) + # of pages written (after merging) or number of passes * # of pages * (read 1 + write 1)
Cost of sorting B = # of pages read (to sort)+ # of pages written (after sorting) + # of pages read (to merge) + # of pages written (after merging) or number of passes * # of pages * (read 1 + write 1)
Cost of joining A+B = # of pages A + # of pages B
Total cost = Cost of Sorting A + Cost of Sorting B + Cost of joining A+B
Cost of Sorting A = 2 * 10,000 * 2
Cost of Sorting B = 2 * 1,000 * 2
Cost of Joining A and B = 10,000 + 1,000
Total Cost = 40,000 + 4,000 + 11,000 = 55,000

. d) Hash Join **(1 mark)**

Formulae:
Total Cost = 3 (# of pages A + # of pages B)
Total Cost = 3 (10,000 + 1,000)

. 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. **(1 mark)**

The most optimal cost would be achieved if each relation was only read once. We could do this by storing the smaller relation in memory, reading in the larger relation page by page and we search the smaller relation for each tuple in the larger relation for matching tuples. The buffer space would have to have the entire smaller relation, one page for reading in the larger relation and one page to serve as an output buffer. So the lowest possible I/O cost for joining A and B would be M + N, where M is # of pages A and N is # of pages B, which is total cost = 10,000 + 1,000 = 11,000. To achieve this cost, the buffer space needed would be N + 1 + 1, which is 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 plan utilizing the information given about different indexes in each part.

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

.
Formula:
RF (col = value) = 1/number of distinct A
RF(col > value) = (High(Col) – value) / (High(Col) – Low(Col))
RF (two conditions) = product of 2 RFs
Result_size (single table) = NTuples(R) * P RF$_{\mathbf{i}}$
RF(E.title = "CEO") = 1/10
RF (E.level > 15) = (20-15)/(20-0) = 1/4
RF (two conditions) = (1/10) * (1/4)
Result Size (single table) = 100,000 * (1/10) * (1/4) = 2500

. 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. **(1 mark)**

Cost of sequential scan = 1,000
Formula:
Clustered B+ tree
Total Cost = (Npages(I) + Npages(R)) * RF (two conditions as the index is on both title and level)
N pages(R) = # of tuples / # of tuples can be fit in a page
N pages(R) = 100,000 / 100 = 1,000
N pages (I) = 200
Total Cost = (200 + 1,000) * (1/40) = 30, which is significantly less than the number of pages (full table scan). So using clustered B+ tree index is better than sequential scan.

. 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. **(1 mark)**

Cost of sequential scan = 1,000
Formulae:
Unclustered B+ tree
Total Cost = (Npages(I) + NTuples(R)) * RF(level)
Total Cost = (200 + 100,000) * (1/4) = 25,050, which is significantly greater than the number of pages (full table scan). So sequential scan is better than unclustered B+ tree index.

. 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. **(1 mark)**

Cost of sequential scan = 1,000
Formulae:
Unclustered Hash
Total Cost = (NTuples(R)) * RF (title) * 2.2
Total Cost = (100,000) * (1/10) * 2.2 = 22,000, which is significantly greater than the number of pages (full table scan). So using sequential scan is better than unclustered hash index.

. 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. **(1 mark)**

Cost of sequential scan = 1,000
Formulae:
Unclustered Hash
Total Cost = (NTuples(R)) * RF(level) * 2.2
Total Cost = 100,000 * (1/4) * 2.2 = 55,000, which is significantly greater than the number of pages (full table scan). So using sequential scan is better than unclustered hash index.

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';*

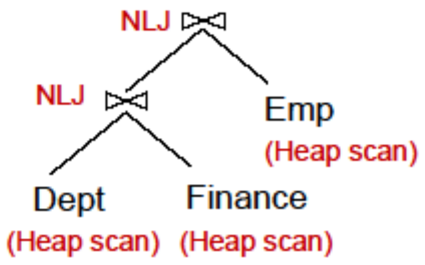
The system's statistics indicate that employee salaries range from 10,000 to 60,000, and employees enjoy 200 different hobbies. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial record in the Finance relation) in the database. Each relation fits 100 tuples in a page. Suppose there exists 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 **(2 marks)**

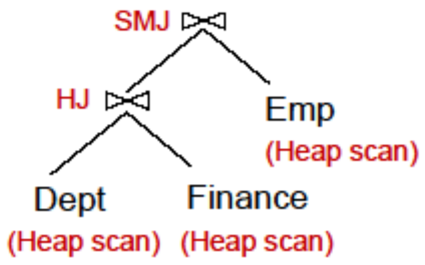
Formula:
RF (col = value) = 1/number of distinct A
RF(col > value) = (High(Col) – value) / (High(Col) – Low(Col))
RF (two conditions) = product of 2 RFs
Result Size (Single Table) = NTuples(R) * P RF_i
RF (E.hobby = 'yodeling') = 1/200
RF (E.sal >= 59,000) = (60,000 – 59,000)/(60,000-10,000) = 1/50
RF (E.hobby = 'yodeling' AND E.sal >= 59,000) = 1/10,000
Result Size = 50,000 * (1/10,000) = 5

. b) 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*: 1st pass to produce partitions, 2nd 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. **(8 marks, 2 marks per plan)**

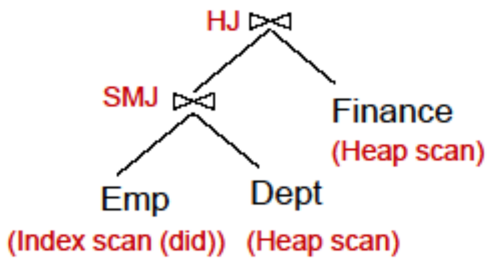
1)



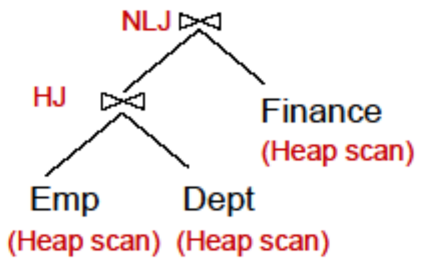
2)



3)



4)



1)
Formula:
# of resulting tuples for one relation join another = 1/keys(I) * NTuples(R) * NTuples(S)
# of resulting pages for one relation join another = # of resulting tuples for one relation join another / # of tuples can fit in a page
Cost of scanning (Nested Loop Join) = # of pages
Cost to join with another relation (Nested Loop Join) = # of pages A * # of pages B
Total Cost = Cost to join with another relation + Cost to join with another relation + Cost of scanning
Assume the number of departments are all distinct
# of resulting tuples = (1/5000) * 5,000 * 5,000 = 5,000
# of resulting pages = 5,000 / 100 = 50
Cost of scanning = 50
# of pages finance = 5000 / 100 = 50
# of pages emp = 50,000 / 100
Cost to join with finance = 50 * 50 = 2,500
Cost to join with emp = 500 * 50 = 25,000
Total Cost = 2,500 + 50 + 25,000

2)
Formula:
# of resulting tuples for one relation join another = 1/keys(I) * NTuples(R) * NTuples(S)

# of resulting pages for one relation join another = # of resulting tuples for one relation join another / # of tuples can fit in a page
Cost to join with another relation (Hash Join) = 2(# of pages in outer) * 3(# of pages B in inner)
Cost of sorting a relation (Sort Merge Join) = # of passes * # of pages * 2. 2 is to read and write.

Cost of joining relations (Sort Merge Join) = # of pages (S) + # of pages (R)
Total Cost (Sort Merge Join) = Cost of sorting a relation + Cost of sorting a relation + Cost of Joining these relations
Total Cost = Total Cost (Sort Merge Join) + Cost to join with another relation (Hash Join)
# of resulting tuples for Dept Join Finance = (1/5,000) * 5,000 * 5,000 = 5,000
# of pages for Dept Join Finance = 5,000 / 100 = 50
Cost to join with Finance (Hash) = 2 (50) + 3 (50) = 250
Number of Passes = 2
Cost of Sorting Dept Join Finance = 2 * 50 * 2 = 200
Cost of Sorting Emp = 2 * 500 * 2 = 2,000
Cost of Joining these together = 50 + 500 = 550
Total Cost of SMJ between Dept Join Finance and Emp = 2,000 + 200 + 550 = 2,750
Total Cost = 2750 + 250 = 3,000

3)
Formula:
# of resulting tuples for one relation join another = 1/keys(I) * NTuples(R) * NTuples(S)
# of resulting pages for one relation join another = # of resulting tuples for one relation join another / # of tuples can fit in a page
Cost of sorting a relation (Sort Merge Join) = # of passes * # of pages * 2. 2 is to read and write.
Cost of joining relations (Sort Merge Join) = # of pages (S) + # of pages (R)
Total Cost (Sort Merge Join) = Cost of sorting a relation + Cost of sorting a relation + Cost of Joining these relations
Cost to join with another relation (Hash Join) = 2(# of pages in outer) * 3(# of pages B in inner)
Total Cost = Total(Sort Merge Join) + Cost to join with another relation (Hash Join)
# of pages emp = 50,000 / 100 = 500
# of pages dept = 5,000 / 100 = 50
# of pages finance = 5,000 / 100 = 50
# of resulting tuples for emp join dept = (1/5,000) * 50,000 * 5,000 = 50,000
# of resulting pages for emp join dept = 50,000 / 100 = 500
Cost of joining sorted Dept and Emp = 50 + 500 = 550
# of passes = 2
Cost of Sorting dept = 2 * 50 * 2 = 200
Cost of Sorting emp = 0. As emp is index scan and already sorted.
Total Cost of SMJ between Dept and Emp = 550 + 200 = 750
Cost to join with Finance (Hash Join) = 2(500) * 3(50) = 1,150
Total Cost = 1150 + 750 = 1,900

4)
Formula:
# of resulting tuples for one relation join another = 1/keys(I) * NTuples(R) * NTuples(S)
# of resulting pages for one relation join another = # of resulting tuples for one relation join another / # of tuples can fit in a page
Cost of scanning (Nested Loop Join) = # of pages
Cost to join with another relation (Nested Loop Join) = # of pages A * # of pages B
Total Cost (Nested Loop Join) = Cost to join with another relation + Cost to join with another relation + Cost of scanning
Cost to join with another relation (Hash Join) = 2(# of pages in outer) * 3(# of pages B in inner)
Total Cost = Total Cost (Nested Loop Join) + Cost to join with another relation (Hash Join)

# of pages emp = 50,000 / 100 = 500
# of pages dept = 5,000 / 100 = 50
# of pages finance = 5,000 / 100 = 50
Cost to join with another relation (Hash Join) = 2(500) + 3(50) = 1150
Cost of scanning Emp Join Dept = 50,000 / 100 = 500
# of resulting tuples for Emp Join Dept = (1/5,000) * 50,000 * 5,000 = 50,000
Cost to join with finance (Nested Loop Join) = 500 * 50 = 25,000
Total Cost = 1,150 + 500 + 25,000 = 26,650