Database Systems 2020, Assignment 3

Name: Aoi Fujii

Question 1 (5 marks) Consider two relations A and B. A has 80,000 tuples, and B has 100,000 tuples. Both relations store 100 tuples per page. Consider the following SQL statement: SELECT * FROM A INNER JOIN B ON A.a = B.a; We wish to evaluate an equijoin between A and B, with an equality condition A.a = B.a. There are 102 buffer pages available for this operation. Both relations are stored as (unsorted) heap files. Neither relation has any indexes built on it. Consider the alternative join strategies described below and calculate the cost of each alternative. Evaluate the algorithms using the number of disk I/O's (i.e., pages) as the cost. For each strategy, provide the formulae you use to calculate your cost estimates.

Let AP be the number of pages in A and BP be the number of pages in B. Then,

AP = 80000 tuples / 100 tuples per page = 800 pages

BP = 100000 tuples / 100 tuples per page = 1000 pages

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

Total cost = (# of pages outer) + (# of pages in outer * # of pages in innter)

= AP + (AP * BP)

= 800 + (800 * 1000)

= 800800 I/Os

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

# of blocks = ceil(# of pages in outer / (# of buffer pages - 2))

= ceil(AP / (102 - 2))

= 8

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

= AP + 8*BP

= 800 + 8 * 1000

= 8800 I/Os

c) Sort-Merge Join. Assume that Sort-Merge Join can be done in 2 passes. (1 mark)

Cost of sorting A = 2 * # of passes * # of pages of A

= 2 * 2 * 800

= 3200 I/Os

Cost of sorting B = 2 * # of passes * # of pages of B

= 2 * 2 * 1000

= 4000 I/Os

Cost of merging A and B = # of pages read of A + # of pages read of B

= AP + BP

= 800 + 1000

= 1800 I/Os

Total cost = Cost of sorting A + Cost of sorting B + Cost of merging A and B

= 3200 + 4000 + 1800

= 9000 I/Os

d) Hash Join (1 mark)

Total cost = 3 * (# of pages in A + # of pages in B)

= 3 * (AP + BP)

= 3 * (800 + 1000)

= 5400 I/Os

e) What would be the lowest possible cost to perform this query, assuming that no indexes are built on any of the two relations, and assuming that sufficient buffer space is available? What would be the minimum buffer size required to achieve this cost? Explain briefly. (1 mark)

The optimal cost would be achieved if each relation was read only once.

Total cost = # of pages in A + # of pages in B

= AP + BP

= 800 + 1000

= 1800 I/Os

The buffer pool would have to store the entire smaller relation, one page for reading in the larger relation and one page for an output buffer so that the matching tuples could be searched from the smaller relation for each tuple in the larger relation.

The minimum number of buffer pages for this cost is min{AP, BP} + 1 + 1 = 802

Question 2 (5 marks) Consider a relation with the following schema: JobSeekers(id, firstname, lastname, city, soughtsalary) The JobSeekers relation consists of 10,000 pages. Each page stores 100 tuples. The online software works in the 8 largest Australian cities and soughtsalary can have values between 60,000 and 160,000 (i.e., [60,000 – 160,000].) Suppose that the following SQL query is executed frequently using the given relation: SELECT * FROM JobSeekers WHERE city = 'Melbourne' AND soughtsalary > 80,000; Your job is to:

RF(a): reduction factor of a

NTuples(R): number of tuples of R

NPages(R): number of pages of R

High(I): maximum value in I

Low(I): lowest value in I

NKeys(I): number of distinct key values for I

a) Compute the reduction factors and the estimated result size in number of tuples. (1 mark)

The reduction factor of sought salary is

RF(sal) = (High(I) - value) / (High(I) – Low(I))

= (160000 - 80000) / (160000 - 60000)

= 4 / 5

The reduction factor of city is

RF(city) = 1 / NKeys(I) = 1 / 8

The number of tuples of Jobseeker is

NTuples(J) = 10000 pages * 100 tuples per page = 1000000 tuples

Result size = NTuples(J) * Product of all reduction factors

= NTuples(J) * RF(sal) * RF(city)

= 1000000 * (4 / 5) * (1 / 8)

= 100000 tuples

b) Compute the estimated cost in number of disk I/O's of the best plan if a clustered B+ tree index on (city, soughtsalary) is the only index available. Suppose there are 2,000 index pages. Discuss and calculate alternative plans. (1 mark)

There are two possible access paths for this query:

- The clustered B+ tree index on (city, soughtsalary), with cost

Cost = product of RFs of matching conditions * (NPages(J) + NPages(I))

= RF(sal) * RF(city) * (10000 + 2000)

= (4 / 5) * (1 / 8) * 12000

= 1200 I/Os

- Full table scan, with cost 10000 I/Os (NPages(J))

Since other indexes are not applicable here, the cheapest access path is the clustered B+ tree index with cost 1200 I/Os.

c) Compute the estimated cost in number of disk I/O's of the best plan if an unclustered B+ tree index on (soughtsalary) is the only index available. Suppose there are 2,000 index pages. Discuss and calculate alternative plans. (1 mark)

There are two possible access paths for this query:

- The unclustered B+ tree index on (soughtsalary), with cost

  Cost = product of RFs of matching conditions * (NTuples(J) + NPages(I))

  = RF(sal) * (1000000 + 2000)

  = (4 / 5) * 1002000

  = 801600 I/Os

- Full table scan, with cost 10000 I/Os (NPages(J))

Since other indexes are not applicable here, the cheapest access path is the full table scan with cost 10000 I/Os.

d) Compute the estimated cost in number of disk I/O's of the best plan if an unclustered Hash index on (city) is the only index available. Discuss and calculate alternative plans. (1 mark)

There are two possible access paths for this query:

- The unclustered Hash index on (city), with cost

  Cost = product of RFs of matching conditions * hash lookup cost *NTuples(J)

  = RF(city) * 2.2 * NTuples(J)

  = (1 / 8) * 2.2 * 1000000

  = 275000 I/Os

- Full table scan, with cost 10000 I/Os (NPages(J))

Since other indexes are not applicable here, the cheapest access path is the full table scan with cost 10000 I/Os.

e) Compute the estimated cost in number of disk I/O's of the best plan if an unclustered Hash index on (soughtsalary) is the only index available. Discuss and calculate alternative plans. (1 mark)

Hash index cannot be used over a range (on soughtsalary to select soughtsalary > 80,000).

Thus, the only option is the full table scan with cost 10000 I/Os.

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 > 100,000 AND E.hobby IN ('diving', 'soccer'); The system's statistics indicate that employee salaries range from 50,000 to 150,000, and employees enjoy 50 different hobbies. There is a total of 25,000 employees and 1,200 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 (Dept.did) and a clustered B+ tree index on (Emp.salary), both of size 50 pages.

Let's denote the number of tuples of emp, dept, finance as NTuples(E) = 25000, NTuples(D) = 1200, NTuples(F) = 1200.

a) Compute the reduction factors and the estimated result size in number of tuples. (2 marks)

The reduction factor of did for E.did = D.did is

RF(didED) = 1 / (Max(NKeys(E.did), NKeys(D.did))) = 1 / NKeys(did) = 1 / 1200

The reduction factor of did for D.did = F.did is

RF(didDF) = 1 / (Max(NKeys(D.did), NKeys(F.did))) = 1 / NKeys(did) = 1 / 1200

The reduction factor of salary is

RF(sal) = (High(sal) - value) / (High(sal) – Low(sal))

= (150000 – 10000) / (150000 - 50000)

= 1 / 2

The reduction factor of hobby is

RF(hob) = 2 / Nkeys(hob) = 2 / 50 = 1 / 25

(selecting diving and soccer out of 50 hobbies)

Result size = NTuples(E) * NTuples(D) * NTuples(F)

* RF(didED) * RF(didDF) * RF(sal) * RF(hob)

= 25000 * 1200 * 1200 * (1 / 1200) * (1 / 1200) * (1 / 2) * (1 / 25)

= 500 tuples

b) Compute the cost in number of disk I/O's of the plans shown below. Assume that sorting of any relation (if required) can be done in 2 passes. NLJ is a Page-oriented Nested Loops Join. Assume that did is the candidate key, and that 50 tuples of a resulting join between Emp and Dept fit in a page. Similarly, 50 tuples of a resulting join between Finance and Dept fit in a page. Any selections/projections not indicated on the plan are performed "on the fly" after all joins have been completed. (8 marks, 2 marks per plan)

Let's denote the number of pages of emp, dept, finance as NP(E) = 25000 / 100 = 250, NP(D) = 1200 / 100 = 12, NP(F) = 1200 / 100 = 12.

1)

Number of resulting tuples for Dept JOIN Finance

= RF(didDF) * NTuples(D) * NTuples(F)

= (1 / 1200) * 1200 * 1200

= 1200 tuples

Number of pages for Dept JOIN Finance

= 1200 tuples / 50 tuples per page

= 24 pages

Cost of scanning Dept = NP(D) = 12

Cost to join with Finance = NP(D) * NP(F) = 12 * 12 = 144

Cost to join with Emp = NP(D JOIN F) * NP(E) = 24 * 250 = 6000

Total cost

= Cost of scanning Dept + Cost to join with Finance + Cost to join with Emp

= 12 + 144 + 6000

= 6156 I/O

2)

Number of resulting tuples for Dept JOIN Finance

= RF(didDF) * NTuples(D) * NTuples(F)

= (1 / 1200) * 1200 * 1200

= 1200 tuples

Number of pages for Dept JOIN Finance

= NP(D JOIN F)

= 1200 tuples / 50 tuples per page

= 24 pages

Cost to Hash join Dept and Finance = 3 * NP(D) + 3 * NP(F) = 3 * 12 + 3 * 12 = 72

Cost to join with Emp:

Cost of sorting (D JOIN F)

= 2 * NPass * NP(D JOIN F) - NP(D JOIN F)

(subtract the number of pages because of the pipelining of D JOIN F)

= 2 * 2 * 24 - 24

= 72

Cost of sorting (Emp) = 2 * NPass * NP(E) = 2 * 2 * 250 = 1000

Cost of joining to Emp = NP(D JOIN F) + NP(E) = 24 + 250 = 274

Cost of MSJ to Emp

= Cost of sorting (D JOIN F) + Cost of sorting (Emp) + Cost of joining to Emp

= 72 + 1000 + 274

= 1346

Total cost = Cost to Hash join + Cost of MSJ = 72 + 1346 = 1418 I/O

3)

Number of resulting tuples for Emp JOIN Dept

= RF(didED) * NTuples(E) * NTuples(D)

= (1 / 1200) * 25000 * 1200

= 25000 tuples

Number of pages for Emp JOIN Dept = NP(E JOIN D) = 25000 / 50 = 500 pages

Cost of sorting Emp = 2 * NPasses * NP(E) = 2 * 2 * 250 = 1000

Cost of sorting Dept = 2 * NPasses * NP(D) = 2 * 2 * 12 = 48

Cost of joining sorted Emp and Dept = NP(E) + NP(D) = 250 + 12 = 262

Cost of SMJ Emp and Dept

= Cost of sorting Emp + Cost of sorting Dept + Cost to join sorted Emp & Dept

= 1000 + 48 + 262 = 1310

Cost to join with Finance

= 2 * NP(E JOIN D) + 3 * NP(F) (by pipelining of E JOIN D)

= 2 * 500 + 3 * 12

= 1036

Total cost = Cost of SMJ + Cost to join with Finance = 1310 + 1036 = 2346 I/O

4)

Number of tuples selected

= NTuples(E selected) = RF(sal) * NTuples(E) = (1 / 2) * 25000 = 12500 tuples

Number of pages of selected tuples

= NP(Emp selected) = 12500 tuples / 100 tuples per page = 125 pages

Cost of selection for E.sal > 100000 using clustered B+ tree index

= (NP(Index) + NP(E)) * product of all the RFs for each predicate

= (50 + 250) * RF(sal)

= 300 / 2

= 150

Number of resulting tuples for selected Emp JOIN Dept

= RF(didED) * NTuples(E selected) * NTuples(D)

= (1 / 1200) * 12500 * 1200

= 12500 tuples

Number of pages for selected Emp JOIN Dept

= NP(selected Emp JOIN Dept) = 12500 tuples / 50 tuples per page = 250 pages

Cost of Hash join with Dept

= 2 * NP(Emp selected) + 3 * NP(D) (due to the pipelining of Emp selected)

= 2 * 125 + 3 * 12 = 286

Cost to join with Finance

= NP(selected Emp JOIN Dept) * NP(F)

= 250 * 12

= 3000

Total cost

= Cost of selection for E.sal + Cost of Hash join + Cost to join with Finance

= 150 + 286 + 3000

= 3436 I/O