Kamyar Karimifakhr 1068176

Assignment 3

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.

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

Cost(PNLJ) =
= NPages(Outer) + ((NPages(Outer))*(NPages(Inner)))
800 + (800*1000) = 800800 I/Os

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

NBlocks(Outer)
= ($NPages(Outer)$)/($Block\ size$–2) = 800/100 = 8 blocks

Cost(BNLJ) = NPages(Outer) + ((NBlocks(Outer)) * (NPages(Inner)))

= 800 +(8*1000) = 8800 I/OS

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

Sort(A) = 2 * Number Passes * NPages(A)
= 2*2*800 = 32000 I/Os

Sort(B) = 2 * Number Passes * NPages(B)
= 2*2*1000 = 4000 I/Os

Cost(A SMJ B)
= NPages(A) + NPages(B) = 800 + 1000 = 1800

Cost(SMJ) = sort(A) + sort(B) + Cost(A SMJ B) =
3200 + 4000 + 1800 = 9000 I/OS

d) Hash Join

Cost(HJ)
= 2 * NPages(Outer) + 2 * NPages(Inner) + NPages(Outer) + NPages(Inner) = 3 * NPages(Inner) + 3 * NPages(Outer) =
3*800 + 3*100 = 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.

Total cost = Npages(outer) + Npages(inner) (Each relation only read once!)

= 800 + 1000 = 1800 I/Os

Minimum buffer needed = min (Npages(outer), Npages(inner)) +1+1 = 802

Question 2

SELECT * FROM JobSeekers

WHERE city = 'Melbourne' AND soughtsalary > 80,000;

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

RF_{city} = 1/key(I) = 1/8

$RF_{citysoughtsalary}$ = (high – value)/(high-low) = (160000-80000)/100000 = 8/10

Result size = Ntuples(Jobseeker) * product of RFs = 1000000 * 1/8 * 8/10 = 100000

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 2,000index pages. Discuss and calculate alternative plans.

First calculating the reduction factors

RF_{city} = 1/key(I) = 1/8

$RF_{soughtsalary} = 8/10$

Cost of a clustered B+ tree index

Cost = Product of RFs of matching condition * (NPages(I) + NPages(Jobseeker))

= 1/8 *8/10 *(2000 + 10000) = 1200 I/Os

Full table scan = Npages = 10000 I/OS

As a result, the Full table scan is a cheaper option.

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.

Cost = Product of RFs of matching condition * (NPages(I) + NTuples(Jobseeker)) = 8/10

*(1000000+2000) = 801600 I/Os

Alternative plan: Full table scan = 10000 I/Os is cheapest option

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.

Cost of unclustered Hash index:

Cost = product of reduction factors * hash lookup cost * Ntuples(Jobseeker)

= 8/10 * 2.2 * (100*10000) = 1760000 I/Os

Alternative plan : Full table scan = 10000 which is a the best plan here

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.

Best plan if an unclustered Hash index on city is the only index available.

Cost = product of reduction factors * hash lookup cost * Ntuples(Jobseeker)

= 1/8 * 2.2 * (100*10000) = 275000 I/Os

Alternative plan:

Full table scan = 10000 I/Os is the cheaper option

Question 3

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.

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

Total tuple size = Ntuple(Emp) *Ntuple(Dep)*Ntuple(Fin)

= 25000*1200*1200 = 36000000000

First we calculate all the reduction factors

RF(E.did = D.did) = 1/25000

RF(D.did = F.did) = 1/1200

RF(E.sal > 100,000) =

(highest – value)/(highest - lowest) = 150000-100000/150000-50000 = 50000/100000 = 0.5

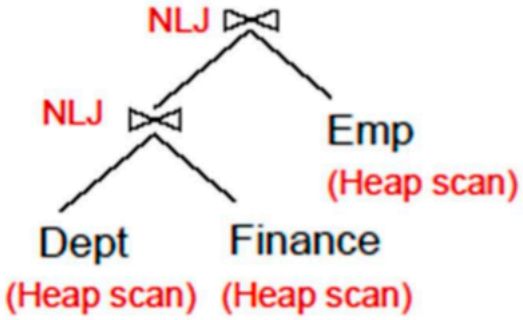
RF(E.hobby IN ('diving', 'soccer'))= 2/52 = 1/26

ESTIMATION :

Ntuples in total * Product of RFs =

36000000000 * 1/25000 * 1/1200 * 0/5 * 1/26 = 23 Tuples

1)



Size of (D NLJ F) = Ntuples(D) * Ntuples(F) * 1/key(I)
= 1200 *1200*1/1200 = 1200 /50 = 24 pages

D scanning cost = 1200/50 = 24 I/Os

Cost of (D NJL F) = Npages(D) + (Npages(d) * Npages(F) = 24 + 24*12 = 312 I/Os

We do not need to (D NJL F) as we already read that so we remove it from (NLJ E) calculation.

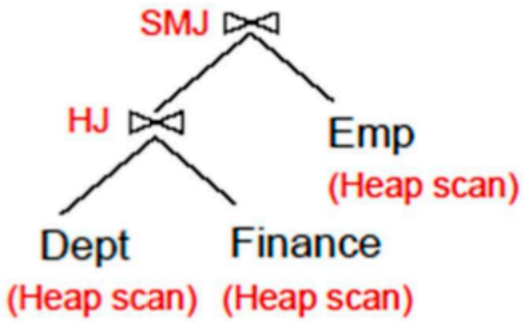
Npages(E) = 25000/100 = 250 pages

Cost to join with employee = Npages(D NJL F)
Npages(E) =

Cost (NJL E) = 24 *250 = 6000 I/Os

Total cost = 312 + 6000 = 6312 I/Os

2)



Ntuples (D HJ F) = Ntuples(D) * Ntuples(F) * 1/key(I) = 1200 * 1200 * 1/1200 = 1200/100 =

1200 Tuples so 1200/50 = 24 pages

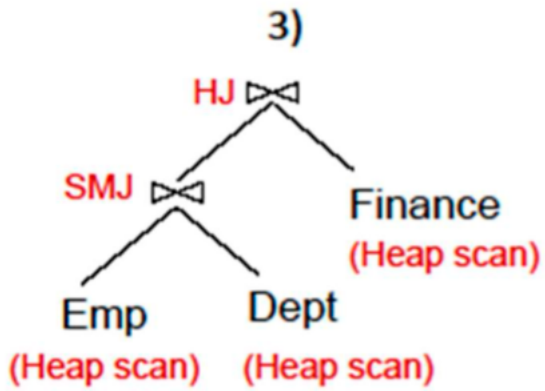
Cost of (D HJ F) = 3 *Npages(Outer) + 3*Npages(Inner) = 3 *50 + 3*12 = 186 I/Os

Since the result of (D HJ F) has already been stored into memory once, it can be pipelined fo the latter stage

Cost((D HJ F) SMJ E)
= ((2 * NPasses * NPages(D HJ F)) + NPages(E)) = (2*2*24) + 250 = 346 I/Os

Costs in total = 186 + 346 = 532 I/Os

3)



Number of resulting tuples for Emp SMJ Dept = 1/ NKeys(I)×NTuples(Emp) × NTuples(Dept)
= 1/25000 * 25000 * 1200 = 1200 Tuples

Number of pages of (Emp SMJ Dept) = 1200 / 50 = 24 pages

Sort Emp = 2 × NPasses × NPages(Emp) = 2 × 2 × 250= 1000 I/O

Sort Dept = 0 (As it is already sorted since it's in a clustered Index)

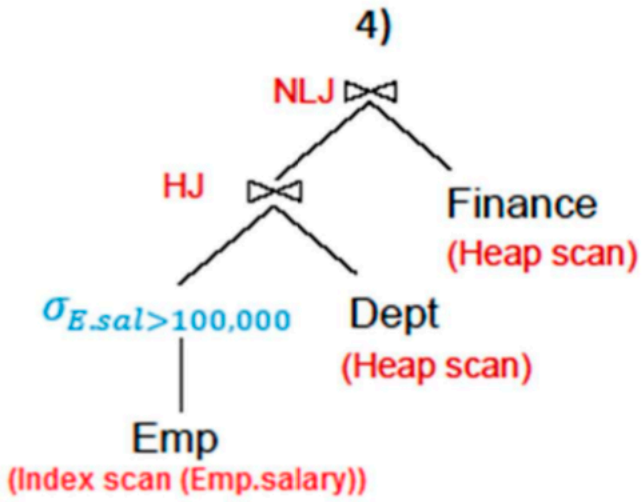
Cost of merge joining Emp and Dept = NPages(Emp) + NPages(Dept)
= 250 + 50 = 300 I/O

Cost of SMJ = 1000 + 300 = 1300 I/Os

We already stored the (E join D) once so we need to multiply it by 2.

Cost of (HJ F) = 2* NPages(Emp JOIN Dept) + 3 * NPages(Finance) = 2* 24 + 3 * 12 = 84 I/Os

Total cost = 1300+ 84 = 1384 I/Os



RF = (High -Value)/(High – Value) = 50000/100000 = 0.5

Selection cost = Npages(I) +Npages(E) * RF

= (50 +250) * 0.5 = 150 I/Os

Number of pages after selection = (rf * Ntuples)/100

(0.5 *25000)/100 = 12500/100 = 125 pages

Size of $Emp_{E.sal>100000}$join Dept = Ntuples($Emp_{E.sal>100000}$) * NTuples(Dept) *1/Nkeys(I) = 12500 *1200 1/12500 *0.5= 600 Tuples =>

Result size = 600/50 = 12 Pages

Cost HJ($Emp_{E.sal>100000}$join Dept) =

3*Npages(Emp) + 3*Npages(Dept) = 3 *125 + 3*50 = 525 I/Os

Npages($Emp_{E.sal>100000}$join Dept) is stored in memory before so no need to calculate again.

Cost (NLJ Finance) = Npages($Emp_{E.sal>100000}$join Dept)* Npages(Fin) = 12*12 = 144 I/Os

Total cost = cost of selection + cost($Emp_{E.sal>100000}$join Dept) +

cost(($Emp_{E.sal>100000}$ $hash$ join Dept)NlJ(Finacne))

Total cost = 150 + 144 + 525 = 819 I/Os