BD2 – April 3rd, 2019 – Solutions – V1.0
Please feel free to answer this test in English, Italian, or any mixture

1. Consider a schema R(IdR, A, B, …, IdT*), S(IdR*, IdT*,C), T(IdT, D, E,…) and the following query

```
SELECT DISTINCT R.IdR, R.A, R.B, R.IdT
FROM            R, S, T
WHERE           R.IdT = S.IdT And S.IdT = T.IdT And S.C = 3
```

Assume that R, S and T are stored as heap files. Primary keys are R.IdR and T.IdT, while R.IdT, S.IdR and R.IdT are foreign keys. The only key for S is the set of its three attributes (S.IdR, S.IdT, S.C). Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on S.C. Assume that the size of all indexes only depend on the number of RIDs, as indicated in the table below. Assume that every page is 4.000 bytes long, and that every attribute uses 4 bytes. Assume a buffer size of 200 pages. If you need Cardenas formula Φ(n,k), approximate it with min(n,k).

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	10.000.000	100.000	20.000			
S	1.000.000	5.000	2.000			
T	100.000.000	1.000.000	400.000			
Idx.S.C			See S	1000	0	100.000

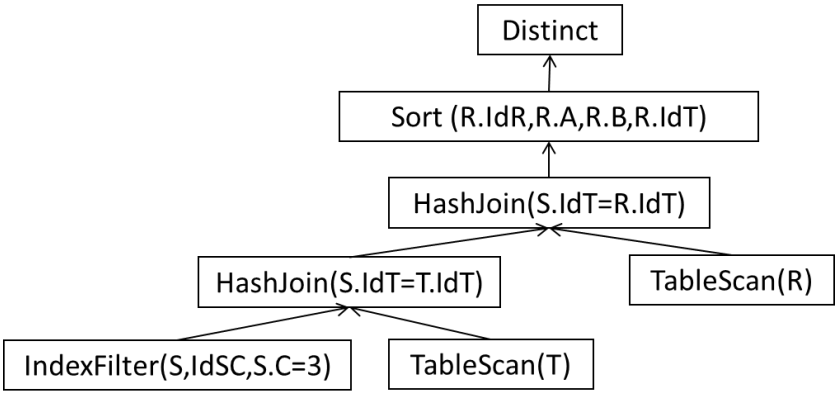
a) Is DISTINCT redundant? Why?

It is not redundant: the closure of {R.IdR, R.A, R.B, R.IdT} with respect to the query is { R.IdR, R.A, R.B, R.IdT, S.IdT, T.IdT, S.C, R.*, T.* } which does not include a key for each relation, in particular, it does not include the S.IdR attribute: observe that we do not have a condition R.IdR=S.IdR in the query.

b) Compute the selectivity factor of the three predicates in the condition

The predicates R.IdT = S.IdT And S.IdT = T.IdT have the same selectivity, 1/NKey(T) = 1/100.000.000.
fs(S.C=3) = 1/NKey(S.C) = 1/1.000

c) Compute the cost of a HashJoin plan with structure HashJoin (HashJoin(IndexFilter(S),T), R) – for simplicity does not use any projection before the end. If you need the size of a record in Join(S,T), you may compute the size of each record of S and T, as 4.000*NPag/NRec, and then do LRec(S)+LRec(T).



C(IndexFilter(S,IdxSC,3,3) = CI + CD = ⌈sf*NLeaf(IdxSC) ⌉ + ⌈Φ(NRec(S)/NKey(C), NPag)⌉ = 2.000/1.000 + ⌈Φ(1.000.000/1.000, 5.000)⌉ = 2 + 1.000 = 1.002
EPag(IndexFilter) = sf(S.C=3)*NPag(S) = (1/1.000)*5.000 = 5 hence, S fits the buffer and the HashJoin has no cost.

C(HashJoin(IndexFilter(S),T)) = C(IndexFilter) + C(TableScan(T)) + 0 = 1.002 + 1.000.000 = 1.001.002

ERec(HashJoin(IndexFilter(S),T) = NRec(S)*NRec(T)* sf(S.C=3)* sf(S.IdT = T.IdT)
= NRec(S)*NRec(T)* sf(S.C=3) / NRec(T) = NRec(S)*sf(S.C=3) = 1.000.000/1.000 = 1.000

LRec(S) = 4000*5000/1.000.000 = 20
LRec(T) = 4000*1.000.000/100.000.000 = 40
LRec(S+T)=60

NPag(HashJoin(S,T)) = NRec(HashJoin)*LRec(HashJoin)/4.000 = 1.000*60/4.000 = 15

Also in this case, the internal relation fits the buffer size, hence the cost of the outermost HashJoin is zero, hence:
C(HashJoin(HashJoin(S,T),R)) = C(HashJoin(S,T)) + C(TableScan(R))
= 1.001.002 + 100.000 + 0 = 1.101.002

ERec(HashJoin(HashJoin(S,T),R)) = ERec(HashJoin(IndexFilter(S),T) * ERec(R) * sf(R.IdT = S.IdT)
= 1.000 * 10.000.000 / 100.000.000 = 100
LRec(R) = 4000*100.000/10.000.000 = 40
LRec(R+S+T) = 100

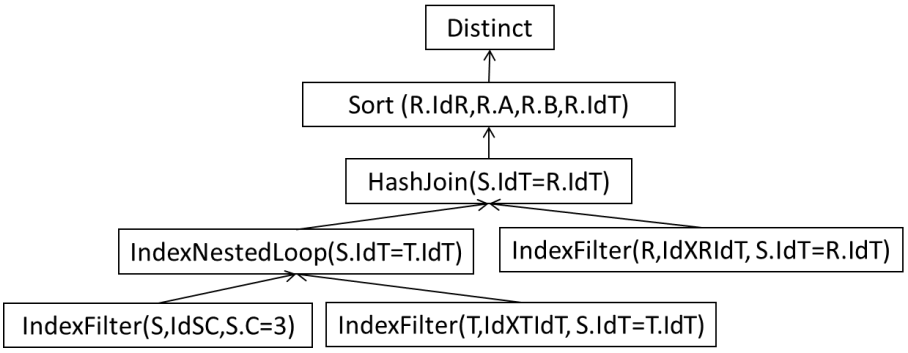
NPag(HashJoin(HashJoin(S,T),R)) = ERec*LRec(R+S+T)/DPag = 100*100 / 4000 = 3
Hence, the Sorthas no cost

C(Distinct) = C(Sort) = C(HashJoin(HashJoin(S,T),R)) = 1.101.002

d) Would the use of projection before the HashJoin reduce their cost? Just explain your answer, but without performing the computation.

One could use the projection in order to optimize the use of the buffer space, but it would not affect the total cost of the access plan.

e) Compute the cost for an IndexNestedLoop plan with the same structure: IndexNestedLoop(IndexNestedLoop (IndexFilter(S),T), R) (draw the plan first)



C(IndexFilter(S,IdxSC,3,3)) = 1.002
C(IndexFilter(T,IdxTIdT,T.IdT=S.IdT) = CI+CD
$= \lceil sf*NLeaf(IdxTIdT) \rceil + 1*\lceil \Phi(NRec(T)/NKey(IdT), NPag(T)) \rceil$
$= \lceil 400.000/100.000.000 \rceil + 1*\lceil \Phi(1, 1.000.000) \rceil = 1+1 = 2$

C(IndexNestedLoop(IndexFilter(S),T)) = C(IndexFilter) + ERec(IndexFilter(S))*C(IndexFilter(T)) =
= 1002 + NRec(S)*sf(S.C=3)*2 = 1002 + (1.000.000/1.000)*2 = 3.002

C(IndexFilter(R,IdxRIdT,T.IdT=S.IdT) = CI+CD
$= \lceil sf*NLeaf(IdxRIdT) \rceil + 1*\lceil \Phi(NRec(R)/NKey(IdT), NPag(R)) \rceil$
$= 1+1*\lceil \Phi(10.000.000/100.000.000, 100.000) \rceil = 2$

ERec(IndexNestedLoop(IndexFilter(S),T)) = 1.000 (as computed in point c)

C(IndexNestedLoop (IndexNestedLoop (S,T)),R)
= C(IndexNestedLoop(S,T)) + ERec(IndexNestedLoop(IndexFilter(S),T))*C(IndexFilter(R))
= 3.002 + 1.000*2 = 5.002

Sort and Distinct have no cost.

f) Which of the two plans is better? Can you explain the main reason why is it cheaper in one sentence (something like: “It is better since it avoids sorting the R3 relation”)?

The second plan is better, since it avoids the complete scan of table T, which is huge. Another acceptable explanation: it is better since it starts from the selected tuples of relation S, which are few, and then it

only retrieves tuples that are related to those in this little set.

2. Consider a relation R(X, Y, A, B, C) that contains 1.000.000 tuples, one for each pair (i,j) such that $0 < i \leq 1000$, $0 < j \leq 1000$. Assume that the relation is stored in a head file, in a completely random order, and that a combined index on (X,Y) is defined on R. The relation occupies 10.000 pages, and the combined index 4.000 pages.
    a. Consider the following condition: $100 < X \leq 110$ AND $100 < Y \leq 110$. How much does it cost to retrieve all records that satisfy it using the index? For simplicity, assume that you first load all relevant RIDs in main memory, you sort them, and you use them to access the records.

We have min(X)=min(Y)=0, max(X)=max(Y)=1.000, NKey(X)=NKey(Y)=1.000.
$Sf(100 < X \leq 110) = Sf(100 < Y \leq 110) = 10/1.000$, hence $sf(100 < X \leq 110 \text{ AND } 100 < Y \leq 110) = 1/10.000$. In practice, we have 10 distinct values for X and 10 distinct values for Y.
Given the structure of the combined index, for each of the 10 values of X we will access a different section of the index, where we retrieve a set of RID which satisfy a condition "X=K and $100 < Y \leq 110$", hence:
$CI = 10*\lceil NLeaf*sf(X=K \text{ and } 100 < Y \leq 110)\rceil = 10*\lceil 4000*(1/1.000*1/100)\rceil = 10*1 = 10$

CD depends on the fact that we sort the RIDs before accessing the data, hence we have just one list:
$CD = \lceil \Phi(sf*NRec(R), NPag(R))\rceil = \lceil \Phi(1.000.000/10.000, 10.000)\rceil = 100.$

CI+CD = 10+100 = 110

    b. How does the cost of point (a) changes if we assume that the index is clustered?

Observe that we have NRec/NPag = 1.000.000/10.000 = 100 record per pages.
With a clustered index, data is clustered wrt X and then wer Y, hance we need to get 10 groups ($10=sf(100 < X \leq 110)*NKey(X)$) of 10 consecutive records ($10=NRec*sf(X=K \text{ and } 100 < Y \leq 110)$)
CI remains 10*1, and CD is also 10*1, since 10 records fit a page:
$\lceil sf(X=K \text{ and } 100 < Y \leq 110)*NPag\rceil = \lceil 10.000 / 100.000\rceil = 1$

. Hence, the total cost is 20 pages.

    c. Assume that a B*-tree, indexed on (X,Y) like the combined index, is used to store the same relation. The B*-tree, keeps all data in the leaves and uses the intermediate nodes for a sparse index. Assuming that the B*-tree has 10.000 leaves, and each intermediate node has space for 500 children, how many intermediate nodes will have the B*-tree? How many levels will it have, leaves included?

The intermediate nodes need to store 10.000 pointers, hence we need 20 nodes. Hence, the B*-tree will have 3 levels: the root, the 16 intermediate nodes, the 10.000 leaves.

    d. How does the cost of point (a) changes if we assume that a B*-tree is used?

We need to get 10 groups of 10 consecutive records., hence we only need to read 10 pages.

e. (This should be a simple question, do not do too many complex calculations) Assume that a G-tree, is used to store the same relation. Like the B*-tree, it keeps all data in the leaves and uses the intermediate nodes for a sparse index. Assuming that it divides the space in a grid of 32000 cells, how many levels would the partition tree have? And the G-tree, how many levels would it have?

Since the data is very regular, the partition tree will be balanced, hence its depth will be:

$\log_2(32000) = 15$.

The G-tree itself will have the same number of nodes and pages as the B*-trees, hence, it will need three levels only.