BD2 – April 9th, 2018 – solutions, V1.0
Please feel free to answer this test in English, Italian, or any mixture

1. Consider a schema R(<u>IdR</u>, A, B, …, IdS*), S(<u>IdS</u>,…), T(<u>IdT</u>, C, D,…, IdS*) and the following query

   SELECT DISTINCT T.C. count(*)
   FROM R, T
   WHERE R.IdS = T.IdS And R.A ≤ 100 And T.D = 10
   GROUP BY T.C, T.D

Assume that R is stored with a dense clustered index on A, while S and T are stored as heap files. Primary keys are R.IdR, S.IdS and T.IdT, while R.IdS and T.IdS are foreign keys that refers to S.
Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on T.D. 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 100 pages. If you need Cardenas formula $\Phi(n,k)$, approximate it with min(n,k).

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	500.000	10.000	1.000			
S	100.000	2.000	200			
T	10.000.000	50.000	20.000			
Idx.R.A			See R	100	0	1.000
Idx.T.D			See T	100	0	1.000

   a) Is DISTINCT redundant?
   b) Compute the selectivity factor of the three predicates in the condition
   c) Compute the cost of accessing $\sigma_{A\leq 100}(R)$, $\sigma_{D\leq 10}(T)$ (useless for this query) and $\sigma_{D=10}(T)$, with and without index. Remember that the index on R.A is clustered.
   d) Compute the cost of a MergeJoin plan that uses the cheapest plans for the selections (ignore the fact that IdS is not a key for neither table) (draw the plan first)
   e) Compute the cost for an IndexNestedLoop plan where R is the outer (left-hand-side) leaf (draw the plan first)

2. a) Assume two relation R(a,b) and S(a,c). Define the left natural outer join(R,S) (leftjoin(R,S)) and give a very simple example.
   b) Write a two-lines (more or less) pseudo code specification for IndexNestedLoop. Briefly specify (using natural language or psudo-code) how one could generalize IndexNestedLoop to LeftIndexNestedLoop in order to compute left outer join
   c) Compare the cost of LeftIndexNestedLoop(OE,OI,condition)) with the cost of IndexNestedLoop(OE,OI),condition)

3. a) Explain (briefly) why heap organization is the one that is most commonly used
   b) In which situations the heap organization is the best one?
   c) In which situations is the hash (procedural) organization the best one?

BD2 – April 9th, 2018 – solutions – 0.1
Please feel free to answer this test in English, Italian, or any mixture

1. Consider a schema R(<u>IdR</u>, A, B, …, IdS*), S(<u>IdS</u>,…), T(<u>IdT</u>, C, D,…, IdS*) and the following query

SELECT DISTINCT T.C. count(*)
FROM R, T
WHERE R.IdS = T.IdS And R.A ≤ 100 And T.D = 10
GROUP BY T.C, T.D

Assume that R is stored with a dense clustered index on A, while S and T are stored as heap files. Primary keys are R.IdR, S.IdS and T.IdT, while R.IdS and T.IdS are foreign keys that refers to S.
Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on T.D.
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 100 pages. If you need Cardenas formula Φ(n,k), approximate it with min(n,k).

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	500.000	10.000	1.000			
S	100.000	2.000	200			
T	1.000.000	50.000	2.000			
Idx.R.A			See R	100	0	1.000
Idx.T.D			See T	100	0	1.000

a) Is DISTINCT redundant?

Yes, it is, since the closure of the projected attribute T.C is {T.C, T.D}, which includes the GROUP BY attributes

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

s_f(R.IdS = T.IdS) = 1/NReg(S) = 1/100.000
s_f(R. A ≤ 100) = 100/Max(R.A) = 100/1.000 = 1/10
s_f(T.D = 10) = 1/NKey(T.D) = 1/100

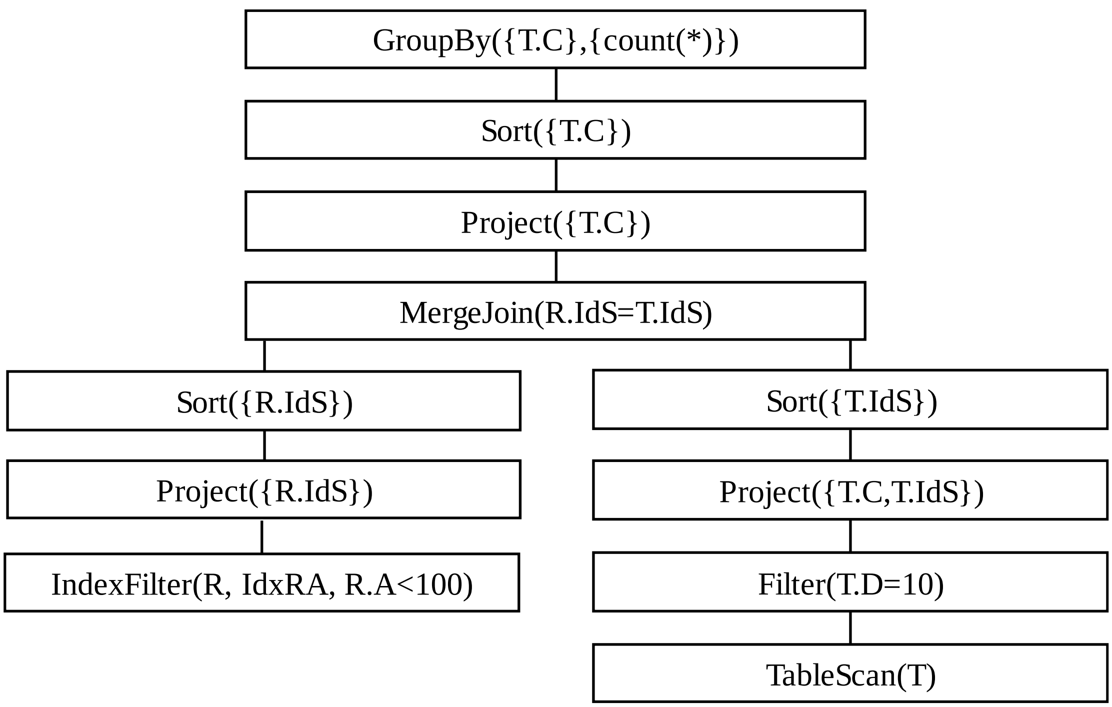
c) Compute the cost of accessing $\sigma_{A\leq100}$(R), $\sigma_{D\leq10}$(T) (useless for this query) and $\sigma_{D=10}$(T), with and without index. Remember that the index on R.A is clustered.

C(Filter(TableScan(R), A≤100)) = NPag(R) = 10.000
C(IndexFilter(R, Idx.R.A, A≤100))
 = CI + CD = s_f(R. A ≤ 100)*NLeaf(Idx.R.A) + s_f(R. A ≤ 100)* NPag(R) = 1/10*1.000 + 1/10*10.000 = 1.100

$C(Filter(TableScan(T), D\leq10)) = NPag(T) = 50.000$
$C(IndexFilter(T, Idx.T.D, D\leq10))$
$= CI + CD$
$= \lceil s_f(T.D\leq10)*NLeaf(Idx.T.D)\rceil + \lceil s_f(T.D\leq10)*NKey(T.D)\rceil * \lceil \Phi(NReg(T)/NKey(T.D),NPag(T))\rceil$
$= 10/1.000*20.000 + 10/1.000*100*min(10.000.000/100,50.000) = 200 + 50.000 = 50.200$

$C(Filter(TableScan(T), D=10)) = NPag(T) = 50.000$
$C(IndexFilter(T, Idx.T.D, D=10))$
$= CI + CD$
$= \lceil s_f(T.D=10)*NLeaf(Idx.T.D)\rceil + \lceil \Phi(NReg(T)/NKey(T.D),NPag(T))\rceil$
$= 1/100*20.000 + min(10.000.000/100, 50.000) = 200 + 50.000 = 50.200$

d) Compute the cost of a MergeJoin plan that uses the cheapest plans for the selections (ignore the fact that IdS is not a key for neither table) (draw the plan first)



After the condition T.D = 10 we can forget the T.D field – this is just a small optimization.

$C(IndexFilter(R, Idx.R.A, A\leq100)) = 1.100$ (see the previous point)
$EReg(IndexFilter(R, Idx.R.A, A\leq100)) = s_f(R.A \leq 100)*NReg(R) = 1/10*500.000 = 50.000$
$EPag(Project(IndexFilter(\ldots),\{R.IdS\}) = 50.000*4/4.000 = 50$
$C(Sort(\{R.IdS\})) = C(IndexFilter) = 1.100$

$C(TableScan(T)) = 50.000$
$EReg(Filter(T.D=10)) = s_f(T.D=10)*NReg(T) = 1/100*10.000.000 = 100.000$

NPag(Project(Filter(…),{T.IdS, T.C}) = 100.000*8/4.000 = 200

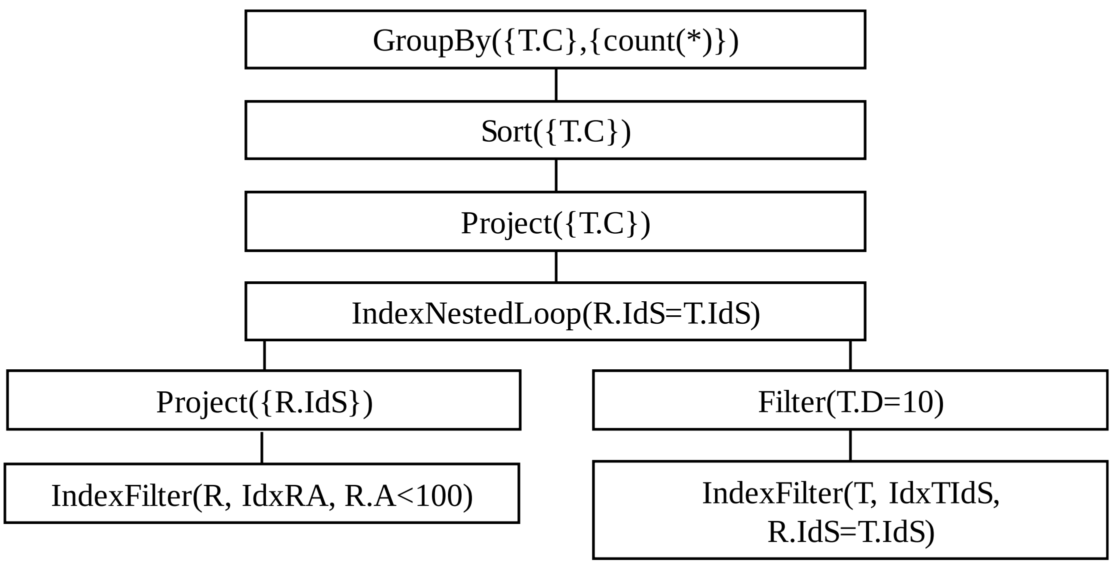
C(Sort(T)) = 2*NPag(Project(Filter(…),{T.IdS, T.C}) + C(TableScan(T)) = 400 + 50.000 = 50.400

C(MergeJoin) = C(Sort…)+C(Sort…) = 1,100 + 50.400 = 51.500

NReg(MergeJoin) = NReg(R)*NReg(T)*s_f(R.IdS = T.IdS)*s_f(R. A $\leq$ 100)*s_f(T.D = 10)
= 500.000*10.000.000/(100.000*10*100) = 50.000
NPag(Project(MergeJoin,{T.C})) = 50.000*4/4.000 = 50

C(Sort(Project(MergeJoin,{T.C})) = C(MergeJoin) = 51.500

e) Compute the cost for an IndexNestedLoop plan where R is the outer (left-hand-side) leaf.



C(IndexFilter(R, Idx.R.A, A$\leq$100)) = 1.100
EReg(IndexFilter(R, Idx.R.A, A$\leq$100)) = 50.000 (see above)

C(IndexFilter(T,…,R.IdS=T.IdS) = CI + CD
= $\lceil s_f(\text{R.IdS=T.IdS})*\text{NLeaf(IdxTIdS)}\rceil + \lceil \Phi(\text{NReg(T)/NKey(T.IdS)},\text{NPag(T)})\rceil$
= $\lceil 20.000/100.000 \rceil + \lceil \Phi(10.000.000/100.000,\ 20.000) \rceil$
=1 + min(100, 20.000) = 101

C(IndexNestedLoop)
= C(OE) + EReg(OE)*C(OI) = 1.100 + 50.000*101 = 1.100 + 5.050.000 = 5.051.100

2. a) Assume two relation R(a,b) and S(a,c). Define the left natural outer join(R,S) (leftjoin(R,S) and give a very simple example.

leftjoin(R,S) contains all the tuple of join(R,S) plus all tuples t of R for which no tuple s exists in S with t.a=s.a, each extended with a null value in the c field.

Example:

R

a	b
1	X
1	Y
2	Z

S

a	c
1	A
1	B
3	C

LeftJoin(R,S)

a	b	c
1	X	A
1	X	B
1	Y	A
1	Y	B
2	Z	null

b) Write a two-lines (more or less) pseudo code specification for IndexNestedLoop. Briefly specify (using natural language or psudo-code) how one could generalize IndexNestedLoop to LeftIndexNestedLoop in order to compute left outer join

```
Index nested loop(R,S):
   for r in R do for s in ( getIndex(S,IdxSa, r.a=s.a) ) do return(r+s);

LeftIndexNestedLoop(R,S):
   for r in R do
      let Set = ( getIndex(S,IdxSa, r.a=s.a) )
      if  Set = empty then return({a=r.a, b=r.b, c=null})
                          else for s in Set do return(r+s);
```

c) Compare the cost of LeftIndexNestedLoop(OE,OI,condition)) with the cost of IndexNestedLoop(OE,OI),condition)

LeftIndexNestedLoop(OE,OI,condition)) only adds some main-memory operations to IndexNestedLoop(OE,OI),condition), hence they have the same cost

3. a) Explain (briefly) why heap organization is the one that is most commonly used
   b) In which situations the heap organization is the best one?
   c) In which situations is the hash (procedural) organization the best one?